






## Article

# Settings-Free Hybrid Metaheuristic General Optimization Methods

Héctor Migallón <sup>1,\*</sup>, Akram Belazi <sup>2,†</sup>, José-Luis Sánchez-Romero <sup>3,†</sup>, Héctor Rico <sup>3,†</sup>  
and Antonio Jimeno-Morenilla <sup>3,†</sup>

<sup>1</sup> Department of Computer Engineering, Miguel Hernández University, E-03202 Elche, Alicante, Spain

<sup>2</sup> Laboratory RISC-ENIT (LR-16-ES07), Tunis El Manar University, Tunis 1002, Tunisia;  
akram.belazi@enit.utm.tn

<sup>3</sup> Department of Computer Technology, University of Alicante, E-03071 Alicante, Spain;  
sanchez@dtic.ua.es (J.-L.S.-R.); hector.rico@gmail.com (H.R.); jimeno@dtic.ua.es (A.J.-M.)

\* Correspondence: [hmigallon@umh.es](mailto:hmigallon@umh.es); Tel.: +34-966658390

† These authors contributed equally to this work.

Received: 2 June 2020; Accepted: 1 July 2020; Published: 3 July 2020



**Abstract:** Several population-based metaheuristic optimization algorithms have been proposed in the last decades, none of which are able either to outperform all existing algorithms or to solve all optimization problems according to the No Free Lunch (NFL) theorem. Many of these algorithms behave effectively, under a correct setting of the control parameter(s), when solving different engineering problems. The optimization behavior of these algorithms is boosted by applying various strategies, which include the hybridization technique and the use of chaotic maps instead of the pseudo-random number generators (PRNGs). The hybrid algorithms are suitable for a large number of engineering applications in which they behave more effectively than the thoroughbred optimization algorithms. However, they increase the difficulty of correctly setting control parameters, and sometimes they are designed to solve particular problems. This paper presents three hybridizations dubbed HYBPOP, HYBSUBPOP, and HYBIND of up to seven algorithms free of control parameters. Each hybrid proposal uses a different strategy to switch the algorithm charged with generating each new individual. These algorithms are Jaya, sine cosine algorithm (SCA), Rao's algorithms, teaching-learning-based optimization (TLBO), and chaotic Jaya. The experimental results show that the proposed algorithms perform better than the original algorithms, which implies the optimal use of these algorithms according to the problem to be solved. One more advantage of the hybrid algorithms is that no prior process of control parameter tuning is needed.

**Keywords:** hybrid optimization algorithms; SCA algorithm; jaya; 2D chaotic map; TLBO; Rao's algorithms

## 1. Introduction

It is well known that metaheuristic optimization methods are widely used to solve problems in several fields of science and engineering. Population-based metaheuristic methods iteratively generate new populations to increase diversity in the current generation. This increases the probability of reaching the optimum for the considered problem. These algorithms are proposed to replace exact optimization algorithms when they are not able to reach an acceptable solution. The inability to provide an adequate solution may be due to either the characteristics of the objective function or the wide search space, which renders a comprehensive search useless. In addition, classical optimization methods, such as greedy-based algorithms, need to consider several assumptions that make it hard to resolve the considered problem.

When metaheuristic methods are operated, on the one hand, the objective function has no restrictions. On the other hand, each optimization method proposes its own rules for the evolution of the population towards the optimum. These algorithms are suitable for general problems, but each one has different skills in global exploration and local exploitation.

Some of the proposed algorithms that have proven to be effective in several areas of science and engineering are studied: mine blast algorithm (MBA) [1] based on the mine bomb explosion concept; the manta ray foraging optimization method (MRFO) [2] based on intelligent behaviors of manta ray; the crow search algorithm (CSA) [3] based on the behavior of crows; the ant colony optimization (ACO) algorithm [4] which imitates the foraging behavior of ant colonies; the biogeography-based optimization (BBO) algorithm [5] which improves solutions stochastically and iteratively; the grenade explosion method (GEM) algorithm [6] based on the characteristics of the explosion of a grenade; the particle swarm optimization (PSO) algorithm [7] based on the social behavior of fish schooling or bird flocking; the firefly (FF) algorithm [8] inspired by the flashing behavior of fireflies; the artificial bee colony (ABC) algorithm [9] inspired by the foraging behavior of honey bees; the gravitational search algorithm (GSA) [10] based on Newton's law of gravity; and the shuffled frog leaping (SFL) algorithm [11] which imitates the collaborative behavior of frogs; among others. Many of them require configuration parameters that must be correctly tuned according to the problem to be solved, see for example [12]. Otherwise, exploitation and exploration skills can be degraded. If the exploitation capacity degrades, the number of populations generated must be increased, while if the exploration capacity deteriorates, the quality of the solution may worsen.

Other proposed algorithms that have also been shown to be effective in various areas of science and engineering but have no algorithm-specific parameters are: the sine cosine algorithm (SCA) [13] based on the sine and cosine trigonometric functions; the teaching-learning-based optimization (TLBO) algorithm [14] based on the processes of teaching and learning; the supply-demand-based optimization method (SDO) [15] based on both the demand relation of consumers and supply relation of producers; the Jaya algorithm [16] based on geometric distances and random processes; the Harris hawks optimization method (HHO) [17] based on the cooperative behavior and chasing style of Harris' hawks, and Rao optimization algorithms [18]; among others.

One of the widely used techniques to improve optimization algorithms is chaos theory. Nonlinear dynamic systems that are characterized by a high sensitivity to their initial conditions are studied in chaos theory [19,20]. They can be applied to replace the PRNGs in producing the control parameters or performing local searches [21–34]. However, improving an optimization algorithm using chaotic systems instead of pseudo-random number generators (PRNGs) may be restricted to the problem under consideration or to a set of problems with similar characteristics.

Hybridization is a well-known strategy that boosts the capacity of optimization algorithms. Since a metaheuristic optimization algorithm cannot overcome all algorithms in solving any problem, hybridization can be a solution that merges the capabilities of different algorithms in one system [35–52]. Many of these algorithms require the correct setting of control parameters, and merging several of these algorithms into a single solution increases the complexity of accurate adjustment of control parameters. Furthermore, some hybridization techniques can be complicated if the management and replacement strategies of individuals in the populations are not similar. On the other hand, when chaos is applied, hybrid algorithms can provide excellent performance for a limited number of applications.

The proposed algorithms consist of hybridizations of seven of the best optimization algorithms that satisfy two requirements: (i) they must be free of algorithm-specific control parameters, and (ii) population management should allow hybridization not only at the population level but also at the individual level.

The remainder of this paper is organized as follows. Section 2 presents a brief description of the optimization algorithms used for the hybridizations. Section 3 describes the hybrid algorithms in detail, analyses of which are provided in Section 4. Finally, concluding remarks are drawn in Section 5.

## 2. Preliminaries

As mentioned above, among the best free control parameter algorithms are the Jaya algorithm [16], the SCA algorithm [13], the supply-demand-based optimization method [15], Rao's optimization algorithms [18], the Harris hawks optimization method (HHO) [17], and the teaching-learning-based optimization (TLBO) algorithm [14]. Among these proposals, the HHO algorithm is the most complex. It consists of two phases. During the first phase, the elements of the population are replaced without comparing the fitness of the associated solutions, which is an unwanted strategy for hybrid algorithms. In addition, the SDO algorithm, which offers impressive initial results, works with two populations preventing its integration in our hybrid proposals.

The Jaya optimization algorithm and the three new Rao's optimization algorithms (i.e., RAO1, RAO2, and RAO3) are described in Algorithm 1. The Jaya optimization algorithm has been successfully used for solving a large number of large-scale industrial problems [53–59]. The three new Rao's optimization algorithms are metaphor-less algorithms based on the best and worst solutions obtained during the optimization process and the random interactions between the candidate solutions [60–62]. In Algorithms 1–3 and 5–9,  $max\_ITs$  is the number of generations;  $popSize$  is the number of individuals in population  $Pop$ ;  $numDesignVars$  is the number of variables of the objective function  $F$ ;  $Pop_m$  is the  $m$ th individual in the current population;  $MinValue^k$  and  $MaxValue^k$  are the low and high bounds of the  $k$ th variable of  $F$ , respectively;  $BestPop$  and  $WorstPop$  are the best and worst individuals of the current population,  $Pop$ , successively;  $newPop_m$  is the  $m$ th new individual that can replace the current  $m$ th individual of population  $Pop_m$ ; and  $rand_{0..1}$  is a uniformly distributed random number in  $[0, 1]$ .

The SCA algorithm, presented in Algorithm 2 has been proven to be efficient in several applications [63–69].

The TLBO algorithm, described in Algorithm 3, is a two-phase algorithm; teacher phase and learner phase. It has been proven effective in solving various engineering problems [70–76].

As mentioned earlier, the use of chaotic maps can improve the behavior of some metaheuristic methods. The 2D chaotic map reported in [33] has significantly improved the convergence rate of the Jaya algorithm [33,77]. The generation of the 2D chaotic map is shown in Algorithm 4, where the initial conditions are  $chA_1 = 0.2$ ,  $chB_1 = 0.3$ ,  $k = i$ , and  $dimMap = 500$ . The computed values of  $chA_i$  and  $chB_i$  are in  $[-1, 1]$ . The chaotic Jaya algorithm (in short, CJaya) is shown in Algorithm 5, where  $ch_x$ ,  $x \in [1 \dots 6]$  are chaotic values randomly extracted from the 2D chaotic map. Other chaotic maps have been applied to Jaya in [32,78]. However, they do not surmount the chaotic behavior of the aforementioned 2D map.

As they present a similar structure, Algorithms 1–5 are used for designing our hybrid algorithms.

**Algorithm 1** Jaya and Rao algorithms

---

```

1: Set  $max\_ITs$  and population size (Iterator individuals:  $m$ )
2: Define the function cost (Iterator design variables:  $k$ )
3: Generate the initial population  $Pop$ 
4: for  $m = 0$  to  $popSize$  do
5:   for  $k = 1$  to  $numDesignVars$  do
6:      $r_1 = rand_{0..1}$ 
7:      $Pop_m^k = MinValue^k + (MaxValue^k - MinValue^k) * r_1$ 
8:   end for
9:   Compute and store function fitness  $F(Pop_m^k)$ 
10: end for
11: for  $iterator = 1$  to  $max\_ITs$  do
12:   Search for the current  $BestPop$  and  $WorstPop$ 
13:   for  $m = 0$  to  $popSize$  do
14:     Select the random individual  $RandPop \neq Pop_m$  {Only for RAO2 and RAO3}
15:     for  $k = 1$  to  $numDesignVars$  do
16:       if Jaya then
17:          $r_1, r_2 = rand_{0..1}$ 
18:          $newPop_m^k = Pop_m^k + r_1 (BestPop^k - |Pop_m^k|) - r_2 (WorstPop^k - |Pop_m^k|)$ 
19:       end if
20:       if RAO1 then
21:          $r_1 = rand_{0..1}$ 
22:          $newPop_m^k = Pop_m^k + r_1 (BestPop^k - WorstPop^k)$ 
23:       end if
24:       if RAO2 then
25:          $r_1, r_2 = rand_{0..1}$ 
26:         if  $Pop_m^k < RandPop^k$  then
27:            $newPop_m^k = Pop_m^k + r_1 (BestPop^k - WorstPop^k) - r_2 (|Pop_m^k| - |RandPop^k|)$ 
28:         else
29:            $newPop_m^k = Pop_m^k + r_1 (BestPop^k - WorstPop^k) - r_2 (|RandPop^k| - |Pop_m^k|)$ 
30:         end if
31:       end if
32:       if RAO3 then
33:          $r_1, r_2 = rand_{0..1}$ 
34:         if  $Pop_m^k < RandPop^k$  then
35:            $newPop_m^k = Pop_m^k + r_1 (BestPop^k - |WorstPop^k|) - r_2 (|Pop_m^k| - |RandPop^k|)$ 
36:         else
37:            $newPop_m^k = Pop_m^k + r_1 (BestPop^k - |WorstPop^k|) - r_2 (|RandPop^k| - |Pop_m^k|)$ 
38:         end if
39:       end if
40:       if  $newPop_m^k < MinValue^k$  then
41:          $newPop_m^k = MinValue^k$ 
42:       end if
43:       if  $newPop_m^k > MaxValue^k$  then
44:          $newPop_m^k = MaxValue^k$ 
45:       end if
46:     end for
47:     if  $F(newPop_m) < F(Pop_m)$  then
48:        $Pop_m = newPop_m$  {replace the current population}
49:     end if
50:   end for
51: end for
52: Search for the current  $BestPop$ 

```

---

**Algorithm 2** SCA optimization algorithm

---

```

1: Set  $iniValue\_r_1 = 2$ 
2: Set  $max\_ITs$  and the population size (Iterator individuals:  $m$ )
3: Define the function cost (Iterator design variables:  $k$ )
4: Generate the initial population  $Pop$  {lines 4–10 of Algorithm 1}
5: for  $iterator = 1$  to  $max\_ITs$  do

6:   Search for the current  $BestPop$ 
7:    $r_1 = iniValue\_r_1 - iterator(iniValue\_r_1 / max\_ITs)$ 
8:   for  $m = 0$  to  $popSize$  do

9:     for  $k = 1$  to  $numDesignVars$  do

10:       $r_2 = 2\pi rand_{0..1}$ ;  $r_3 = 2rand_{0..1}$ ;  $r_4 = rand_{0..1}$ 
11:      if  $r_4 < 0.5$  then
12:         $newPop_m^k = Pop_m^k(r_1 \sin(r_2) \mid r_3 BestPop^k - Pop_m^k \mid)$ 
13:      else
14:         $newPop_m^k = Pop_m^k(r_1 \cos(r_2) \mid r_3 BestPop^k - Pop_m^k \mid)$ 
15:      end if
16:      Check the bounds of  $newPop_m^k$  {lines 40–45 of Algorithm 1}
17:    end for
18:    if  $F(newPop_m) < F(Pop_m)$  then
19:       $Pop_m = newPop_m$  {replace the current population}
20:    end if
21:  end for
22: end for
23: Search for the current  $BestPop$ 

```

---

**Algorithm 3** TLBO algorithm

---

```

1: Set  $iniValue_{r_1} = 2$ 
2: Set  $max\_ITs$  and the population size (Iterator individuals:  $m$ )
3: Define the function cost (Iterator design variables:  $k$ )
4: Generate the initial population  $Pop$  {lines 4–10 of Algorithm 1}
5: Set  $Phase = TeacherPhase$ 
6: for  $iterator = 1$  to  $max\_ITs$  do
7:   Search for the current  $BestPop$ 
8:   Set the teaching factor  $T_F$  (an integer random value  $\in [1, 2]$ )
9:   for  $k = 1$  to  $numDesignVars$  do
10:     $AveragePop^k = (\sum_1^m Pop^k) / numDesignVars$ 
11:  end for
12:  for  $m = 0$  to  $popSize$  do
13:    Select the random individual  $RandPop \neq Pop_m$ 
14:    for  $k = 1$  to  $numDesignVars$  do
15:      if  $Phase = TeacherPhase$  then
16:         $r_1 = rand_{0..1}$ 
17:         $newPop_m^k = Pop_m^k + r_1 (BestPop^k - T_F AveragePop^k)$ 
18:      end if
19:      if  $Phase = LearnerPhase$  then
20:         $r_1 = rand_{0..1}$ 
21:        if  $Pop_m^k < RandPop^k$  then
22:           $newPop_m^k = Pop_m^k + r_1 (Pop_m^k - RandPop^k)$ 
23:        else
24:           $newPop_m^k = Pop_m^k + r_1 (RandPop^k - Pop_m^k)$ 
25:        end if
26:      end if
27:      Check the bounds of  $newPop_m^k$  {lines 40–45 of Algorithm 1}
28:    end for
29:    if  $F(newPop_m) < F(Pop_m)$  then
30:       $Pop_m = newPop_m$  {replace the current population}
31:    end if
32:  end for
33:  if  $Phase = TeacherPhase$  then
34:     $Phase = LearnerPhase$ 
35:  else
36:     $Phase = TeacherPhase$ 
37:  end if
38: end for
39: Search for the current  $BestPop$ 

```

---

**Algorithm 4** 2D chaotic map

---

```

1: Set  $x_1, y_1$  and  $dimMap$ 
2: for  $i = 1$  to  $dimMap$  do
3:    $chA_{i+1} = \cos(k * \arccos(chB_i))$ 
4:    $chB_{i+1} = 16chA_i^5 - 20chA_i^3 + 5chA_i$ 
5: end for

```

---

**Algorithm 5** Chaotic 2D Jaya algorithm

---

```

1: Set  $iniValue_{r_1} = 2$ 
2: Set  $max\_ITs$  and the population size (Iterator individuals:  $m$ )
3: Define the function cost (Iterator design variables:  $k$ )
4: Generate the initial population  $Pop$  {lines 4–10 of Algorithm 1}
5: for  $iterator = 1$  to  $max\_ITs$  do

6:   Search for the current  $BestPop$ 
7:   Search for the current  $WorstPop$ 
8:   Set the scaling factor  $S_F$  (integer random value  $\in [1, 2]$ )
9:   for  $m = 0$  to  $popSize$  do

10:    Select the random individual  $RandPop \neq Pop_m$ 
11:     $r_1, r_2 = rand_{0.1}$ 
12:     $r_a = \min(r_1, r_2)$ 
13:     $r_b = \max(r_1, r_2)$ 
14:    for  $k = 1$  to  $numDesignVars$  do

15:      Extract  $ch_1, ch_2, ch_3, ch_4, ch_5, ch_6$ 
16:      if  $ch_1 < a$  then
17:         $newPop_m^k = ch_2 RandPop_m^k + ch_3 (Pop_m^k - ch_4 RandPop_m^k)$ 
18:         $+ ch_5 (BestPop_m^k - ch_6 RandPop_m^k)$ 
19:      end if
20:      if  $a < ch_1 < b$  then
21:         $newPop_m^k = ch_2 RandPop_m^k + ch_3 (Pop_m^k - ch_4 RandPop_m^k)$ 
22:         $+ ch_5 (WorstPop_m^k - ch_6 RandPop_m^k)$ 
23:      end if
24:      if  $ch_j > b$  then
25:         $newPop_m^k = ch_2 BestPop_m^k + ch_3 (RandPop_m^k - S_F BestPop_m^k)$ 
26:      end if
27:      Check the bounds of  $newPop_m^k$  {lines 40–45 of Algorithm 1}
28:    end for
29:    if  $F(newPop_m) < F(Pop_m)$  then
30:       $Pop_m = newPop_m$  {replace the current population}
31:    end if
32:  end for
33: end for
34: Search for the current  $BestPop$ 

```

---

**3. Hybrid Algorithms**

The proposed hybrid algorithms are designed using the seven algorithms described in Section 2. These algorithms have been selected thanks to their performance in solving constrained and unconstrained functions, but also because they share a similar structure that allows the implementation of different hybridization strategies.

Algorithm 6 shows the skeleton of the proposed hybrid algorithms, which includes all common and uncommon tasks without any updating procedure of the current population. Since the TLBO algorithm is a two-phase algorithm, the proposed hybrid algorithms apply these two phases consecutively to each individual. In contrast to the other algorithm where a single-phase is executed, a control parameter *Phase* is applied to process twice the same individual when the TLBO algorithm is used (see lines 24–29 of Algorithm 6). The algorithm used to obtain a new individual is determined by *AlgSelected* (see line 17 of Algorithm 6). In Algorithms 6–9, *AlgSelected* determines the algorithm accountable for producing a new individual.

Given that only algorithms that are free of control parameters have been considered, proposals that require the inclusion of control parameters have been discarded. Following these guidelines, we have designed three hybrid algorithms, an analysis of which is provided in Section 4. The first proposed hybrid algorithm, shown in Algorithm 7, processes the entire population in each iteration using the same algorithm, and is referred to as the HYBPOP algorithm. This is the most straightforward hybridization technique where the requirement to follow the structure given by Algorithm 6 is not

mandatory on all algorithms. In Algorithms 7–9, *NumOfAlgorithms* is the number of algorithms free of control parameters involved in the hybrid proposals.

---

**Algorithm 6** Skeleton of hybrid algorithms
 

---

```

1: Set  $max\_ITs$  and the population size (Iterator individuals:  $m$ )
2: Define the function cost (Iterator design variables:  $k$ )
3: Set  $iniValue\_r1 = 2$ ;  $Phase = TeacherPhase$ ;  $RepeatTLBO = false$ 
4: Generate the initial population  $Pop$  {lines 4–10 of Algorithm 1}
5: for  $iterator = 1$  to  $max\_ITs$  do

6:   Search for the current  $BestPop$  and  $WorstPop$ 
7:   Set the scaling factor  $S_F$  and teaching factor  $T_F$  (an integer random value  $\in [1, 2]$ )
8:   for  $k = 1$  to  $numDesignVars$  do

9:      $AveragePop^k = (\sum_1^m Pop^k) / numDesignVars$ 
10:   end for
11:    $r_1 = iniValue\_r1 - iterator(iniValue\_r1 / max\_ITs)$ 
12:   for  $m = 0$  to  $popSize$  do

13:     Select random individual  $RandPop \neq Pop_m$ 
14:      $r_2, r_3 = rand_{0..1}$ 
15:      $r_a = \min(r_2, r_3)$ ;  $r_b = \max(r_2, r_3)$ 
16:     for  $k = 1$  to  $numDesignVars$  do

17:        $\Rightarrow (AlgSelected)$  Compute  $newPop_m^k$  using  $AlgSelected$  (one from Algorithms 1–5)
18:       Check the bounds of  $newPop_m^k$  {lines 40–45 of Algorithm 1}
19:     end for
20:     if  $F(newPop_m) < F(Pop_m)$  then

21:        $Pop_m = newPop_m$  {Replace the current population}
22:     end if
23:     if  $TLBO$  then

24:       if  $Phase = TeacherPhase$  then

25:          $Phase = LearnerPhase$ ;  $RepeatTLBO = true$ ;  $m = m - 1$ 
26:       else

27:          $Phase = TeacherPhase$ ;  $RepeatTLBO = false$ 
28:       end if
29:     end if
30:   end for
31: end for
32: Search for the current  $BestPop$ 
  
```

---

**Algorithm 7** HYBPOP: Hybrid algorithm based on population
 

---

```

1:  $NumOfAlgorithms = 7$ 
2:  $ALGS[NumOfAlgorithms] = \{Jaya, Chaotic\ Jaya, SCA, RAO1, RAO2, RAO3, TLBO\}$ 
3:  $Selection = 0$ 
4: for  $iterator = 1$  to  $max\_ITs$  do

5:    $AlgSelected = ALGS[Selection]$ 
6:   for  $m = 0$  to  $popSize$  do

7:     for  $k = 1$  to  $numDesignVars$  do

8:       Compute  $newPop_m^k$  using  $AlgSelected$ 
9:     end for
10:   end for
11:    $Selection = Selection + 1$ 
12:   if  $Selection \geq NumOfAlgorithms$  then

13:      $Selection = 0$ 
14:   end if
15: end for
16: Search for the current  $BestPop$ 
  
```

---



The second algorithm, named HYBSUBPOP, is described through Algorithm 8. It logically splits the population into sub-populations. During the optimization process, each sub-population will be processed by one of the seven algorithms mentioned previously.

---

**Algorithm 8** HYBSUBPOP: Hybrid algorithm based on sub-populations
 

---

```

1: NumOfAlgorithms = 7
2: Split popSize into NumOfAlgorithms sub-populations
3: ALGS[NumOfAlgorithms] = {Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, TLBO}
4: for iterator = 1 to max_ITS do
5:   for m = 0 to popSize do
6:     subPopID = sub-population index of individual m.
7:     AlgSelected = ALGS[subPopID]
8:     for k = 1 to numDesignVars do
9:       Compute newPopmk using AlgSelected
10:    end for
11:  end for
12: end for
13: Search for the current BestPop
  
```

---

Algorithm 9 shows the third proposed hybrid algorithm, dubbed HYBIND, in which a different algorithm in each iteration handles each individual of the population.

---

**Algorithm 9** HYBIND: Hybrid algorithm based on individuals
 

---

```

1: NumOfAlgorithms = 7
2: ALGS[NumOfAlgorithms] = {Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, TLBO}
3: Selection = 0
4: for iterator = 1 to max_ITS do
5:   Selection = iterator%7
6:   for m = 0 to popSize do
7:     AlgSelected = ALGS[Selection]
8:     for k = 1 to numDesignVars do
9:       Compute newPopmk using AlgSelected
10:    end for
11:     Selection = Selection + 1
12:     if Selection ≥ NumOfAlgorithms then
13:       Selection = 0
14:     end if
15:   end for
16: end for
17: Search for the current BestPop
  
```

---

It is worth noting that the aim of the proposed hybrid algorithms is not to improve the convergence ratio of the used algorithms separately, nor to perform optimally for a particular problem. It is to show outstanding performance for a large number of problems without adjusting any control parameters of the considered algorithms.

#### 4. Numerical Experiments

In this section, the performance of the proposed hybrid algorithms is analyzed through solving 28 well-known unconstrained functions (see Table 1), the definitions of which can be seen in [77]. The proposed algorithms were implemented in the C language, using the GCC v.4.4.7 [79], and an Intel Xeon E5-2620 v2 processor at 2.1 GHz. The hybrid proposals, along with the original algorithms, have been implemented and tested using C language. The C implementations of the original algorithms used are not available through the Internet. However, their Java/Matlab implementations are commonly available.

**Table 1.** Benchmark functions. Names and parameters.

Id.	Name	Num. vars (D)	Domain (Min,Max)	Id.	Name	Num. vars (D)	Domain (Min,Max)
F1	Sphere	30	−100,100	F15	Bohachevsky_1	2	−100,100
F2	SumSquares	30	−10,10	F16	Booth	2	−10,10
F3	Beale	2	−4.5,4.5	F17	Michalewicz_2	2	0, $\pi$
F4	Easom	2	−100,100	F18	Michalewicz_5	5	0, $\pi$
F5	Matyas	2	−10,10	F19	Bohachevsky_2	2	−100,100
F6	Colville	4	−10,10	F20	Bohachevsky_3	2	−100,100
F7	Trid 6	6	− $D^2$ , $D^2$	F21	Goldstein-Price	2	−2,2
F8	Trid 10	10	− $D^2$ , $D^2$	F22	Perm	4	− $D$ , $D$
F9	Zakharov	10	−5,10	F23	Hartman_3	3	0,1
F10	Schwefel_1.2	30	−100,100	F24	Ackley	30	−32,32
F11	Rosenbrock	30	−30,30	F25	Penalized_2	30	−50,50
F12	Dixon-Price	5	−10,10	F26	Langermann_2	2	0,10
F13	Foxholes	2	− $2^{16}$ , $2^{16}$	F27	Langermann_5	5	0,10
F14	Branin	2	$x_1$ : −5,10 $x_2$ : 0,15	F28	Fletcher-Powell_5	5	$x_i, \alpha_i$ : − $\pi$ , $\pi$ $a_{ij}, b_{ij}$ : −100,100

The data collected from the experimental analysis are as follows:

- **NoR-AI**: the total number of replacements for any individual.
- **NoR-BI**: the total number of replacements for the current best individual.
- **NoR-BwT**: the total number of replacements for the current best individual with an error of less than 0.001.
- **LtI-AI**: the last iteration (*iterator*) in which a replacement of any individual occurs.
- **LtI-BI**: the last iteration (*iterator*) in which a replacement of the best individual occurs.

Three of the five analyzed data (NoR-) indicate the number of times the current individual ( $Pop_m$ ) is replaced by a new individual ( $newPop_m$ ), which provides a better fitness function (see line 21 of Algorithm 6), while the remaining two (LtI-) refer to the last generation (iterator) in which at least one individual has been replaced.

All data given below have been obtained under 50 runs, 50,000 iterations ( $max\_ITs = 50000$ ) and two population sizes ( $popSize = 140$  and  $210$ ). The maximum values of the analyzed data are listed in Table 2.

**Table 2.** Maximum values of the analyzed data.

Population Size	70	140	210
NoR-AI	3,500,000	7,000,000	10,500,000
NoR-BI	3,500,000	7,000,000	10,500,000
NoR-BwT	3,500,000	7,000,000	10,500,000
LtI-AI	49,999	49,999	49,999
LtI-BI	49,999	49,999	49,999

Tables 3–5 show the data of all the considered algorithms independently, i.e., without hybridization. As expected, the behavior of the different algorithms does not follow a familiar pattern. In addition, it depends on the objective function. Regarding a global convergence analysis, both TLBO and CJaya behave better but with a higher order of complexity (see [77,80]). Moreover, it is noted that when using TLBO,

two new individuals are generated in each iteration; one in the teacher phase and the other one in the learner phase. The values in brackets in Tables 3–5 refer to the standard deviation of the data under 50 runs. Note that heuristic optimization algorithms are partially based on randomness, which leads to high values of standard deviation. The average standard deviations are approximately equal to 16%, 22%, 15%, 30%, 23%, 23%, and 22% for Jaya, Chaotic Jaya, SCA, RAO1, RAO2, RAO3, and TLBO, respectively.

An important aspect, not shown in Tables 3–5, is whether the solution obtained by each algorithm is acceptable or not. In particular, the original algorithms fail to obtain a solution tolerance of less than 0.001 for 3, 8, 2, 4, 7, 5, and 2 functions for Jaya, CJaya, SCA, RAO1, RAO2, RAO3, and TLBO, respectively. Therefore, considering only original algorithms, there is no algorithm whose behavior is always the best, which justifies the development of a generalist hybrid system that can solve a large number of benchmark functions and engineering problems.

Comparing the quality of the solutions obtained from the proposed hybrid algorithms, it can be concluded that the HYBSUBPOP algorithm is the worst one because the same thoroughbred algorithm is always applied to the same sub-population, which degrades the algorithm's performance for a small population. Contrary to HYBSUBPOP, the HYBPOP and HYBIND algorithms apply the selected algorithms to all individuals, which leads to better-exploiting hybridizations. The HYBSUBPOP algorithm fails to obtain a solution tolerance of less than 0.001 in 3 functions (F11, F23, and F27) and the HYBPOP and HYBIND algorithms fail in only one function (F27 and F11, respectively). If the population size is increased to 210, the HYBIND algorithm succeeds with all functions, thus the HYBIND algorithm has a slightly better performance in comparison to HYBPOP.

Local exploration has improved both in the HYBPOP method and especially in the HYBIND method, as stated above. Figures 1 and 2 show the convergence curves of both all the individual methods and the three hybrid methods proposed for the first 1000 and 100 iterations, respectively, for functions F1, F8, F11, and F18. Each point in both figures is the average of the data obtained from 10 runs. As shown in these figures, the curves of the three hybrid methods are similar to the curves of the best single algorithms for each function. Therefore, global exploitation, while not improving all methods, behaves similarly to the best single methods for each function. It should be noted that the hybrid methods behave similarly to the best individual methods for each function, which are not always the same.

Table 6 sorts the algorithms according to the number of iterations required to obtain an error of less than 0.001, if an algorithm is missing in a row an acceptable solution is not reached. As seen from this table, no algorithm outperforms all other algorithms. Moreover, a computational cost analysis would be necessary to classify them correctly. Table 7 exhibits the computational cost of different algorithms. It reveals from this table that the hybrid algorithms are mid-ranked in terms of computational cost, and HYBIND is computationally less expensive than HYBPOP.

An analysis of the contribution of each algorithm in the HYBPOP and HYBIND algorithms is exhibited in Tables 8–10. Table 8 indicates the number of times that an individual has been replaced in each algorithm. The replacement is accepted when the new individual improves the fitness of the current solution. As seen from Table 8, the HYBIND algorithm performs more replacements of individuals. In addition, the numbers of replacements per individual for the contributing algorithms are nearly equal, except for the RAO1 algorithm, where the contribution to replacements is limited. The standard deviations of each data (from 50 runs) are being put in brackets. We found that, on average, the standard deviations for HYBPOP and HYBRID algorithms are both equal to 14%.

Table 9 shows the last iteration in which each optimization algorithm replaces an individual in the population, i.e., when it no longer brings improvement to the hybrid algorithm. As can be seen from Table 8, the optimization algorithms, except the RAO1 algorithm, work efficiently in the hybrid algorithms. It is also revealed that the considered algorithms contribute to more generations in the HYBIND algorithm. The mean value of the standard deviation rises to 28% and 23% for HYBPOP and HYBIND, respectively, due to the randomness behavior and lower LtI-AI costs.

**Table 3.** Analysis of Jaya and chaotic Jaya on function F1-F28 with a population size of 140.

	Jaya					Chaotic Jaya				
	NoR -AI	NoR -BI	NoR -BwT	LtI -AI	LtI -BI	NoR -AI	NoR -BI	NoR -BwT	LtI -AI	LtI -BI
F1	293516 (1323)	3676 (37)	3434 (34)	49999 (0)	49971 (27)	75435 (421)	1580 (58)	1525 (55)	1168 (5)	1157 (5)
F2	294070 (1444)	3653 (75)	3435 (68)	49999 (0)	49990 (8)	75324 (449)	1581 (51)	1526 (50)	1163 (8)	1151 (8)
F3	8957 (107)	70 (7)	60 (7)	862 (164)	778 (163)	2394 (52)	20 (4)	9 (2)	49719 (168)	28810 (8641)
F4	5045 (69)	43 (4)	24 (4)	451 (77)	412 (74)	2453 (41)	26 (5)	8 (2)	49773 (214)	27786 (12758)
F5	96251 (253)	739 (32)	729 (34)	8351 (65)	8236 (65)	1930 (528)	14 (11)	8 (7)	41 (12)	8 (6)
F6	17568 (129)	151 (15)	112 (11)	25293 (1564)	22919 (1752)	3264 (358)	54 (19)	0 (0)	49843 (122)	41426 (11336)
F7	11683 (80)	115 (8)	56 (4)	49044 (1099)	10094 (8036)	3856 (275)	60 (10)	0 (0)	49722 (339)	37363 (7744)
F8	18075 (658)	197 (28)	79 (6)	49358 (561)	28600 (9924)	6030 (743)	114 (23)	0 (0)	49722 (303)	35566 (10334)
F9	249027 (939)	2288 (38)	2206 (40)	49999 (0)	49968 (18)	27271 (12031)	443 (289)	421 (264)	552 (281)	446 (210)
F10	7522 (380)	70 (12)	0 (0)	49993 (6)	48950 (1026)	76978 (1835)	1465 (109)	1411 (109)	1597 (37)	1552 (116)
F11	59193 (7991)	956 (239)	375 (174)	49992 (11)	49660 (773)	9367 (1936)	197 (48)	0 (0)	49925 (47)	43912 (7236)
F12	7648 (891)	74 (11)	27 (10)	41236 (14771)	25124 (18788)	3519 (70)	56 (5)	0 (0)	49765 (320)	34756 (11038)
F13	2658 (73)	23 (3)	0 (0)	49779 (272)	30524 (10879)	3220 (553)	37 (10)	0 (0)	49666 (292)	38222 (9294)
F14	3261 (925)	29 (7)	0 (0)	21719 (18396)	17234 (16324)	2023 (111)	21 (5)	0 (0)	49713 (253)	32070 (11911)
F15	6084 (62)	42 (4)	22 (3)	275 (9)	234 (9)	1639 (258)	21 (12)	8 (3)	30 (2)	11 (6)
F16	2827 (71)	24 (4)	8 (1)	49678 (261)	38126 (10321)	2413 (41)	27 (5)	9 (3)	49709 (227)	34212 (12119)
F17	4945 (52)	36 (6)	1 (0)	4098 (1714)	174 (12)	2134 (47)	18 (4)	1 (0)	49739 (333)	34560 (12597)
F18	9958 (134)	96 (10)	0 (0)	48344 (723)	8329 (1549)	2162 (60)	20 (3)	0 (0)	49648 (320)	36491 (10256)
F19	6247 (53)	46 (6)	25 (6)	329 (20)	273 (18)	1708 (315)	19 (13)	6 (2)	32 (4)	10 (5)
F20	6258 (70)	48 (6)	24 (4)	657 (251)	444 (27)	1597 (217)	16 (9)	4 (1)	31 (2)	7 (2)
F21	2599 (52)	18 (4)	4 (1)	49818 (141)	37023 (10059)	2543 (36)	28 (5)	14 (2)	49651 (241)	27864 (14620)
F22	1841 (176)	15 (3)	0 (0)	48784 (1361)	18724 (10505)	1907 (92)	17 (3)	0 (0)	49850 (123)	24366 (11825)
F23	5976 (84)	51 (6)	0 (0)	394 (81)	180 (9)	2170 (35)	19 (4)	0 (0)	49662 (323)	23570 (12705)
F24	41575 (269)	415 (23)	205 (12)	9044 (5088)	4874 (2387)	6509 (158)	117 (25)	55 (19)	124 (11)	95 (17)
F25	58079 (593)	840 (25)	621 (20)	8652 (111)	8613 (111)	5559 (852)	75 (20)	0 (0)	49810 (255)	40191 (7868)
F26	5063 (110)	43 (5)	0 (0)	13854 (8815)	11630 (5926)	2389 (200)	22 (7)	0 (0)	49848 (116)	32604 (13247)
F27	9071 (452)	79 (9)	23 (25)	25536 (18586)	9665 (3926)	2127 (140)	21 (4)	0 (0)	49698 (284)	31794 (11266)
F28	2119 (159)	15 (2)	0 (0)	49427 (751)	24846 (11088)	2156 (124)	21 (6)	0 (0)	49635 (353)	33067 (12178)

**Table 4.** Analysis of sine cosine algorithm (SCA) and Rao's optimization algorithm 1 (RAO1) on function F1-F28 with a population size of 140.

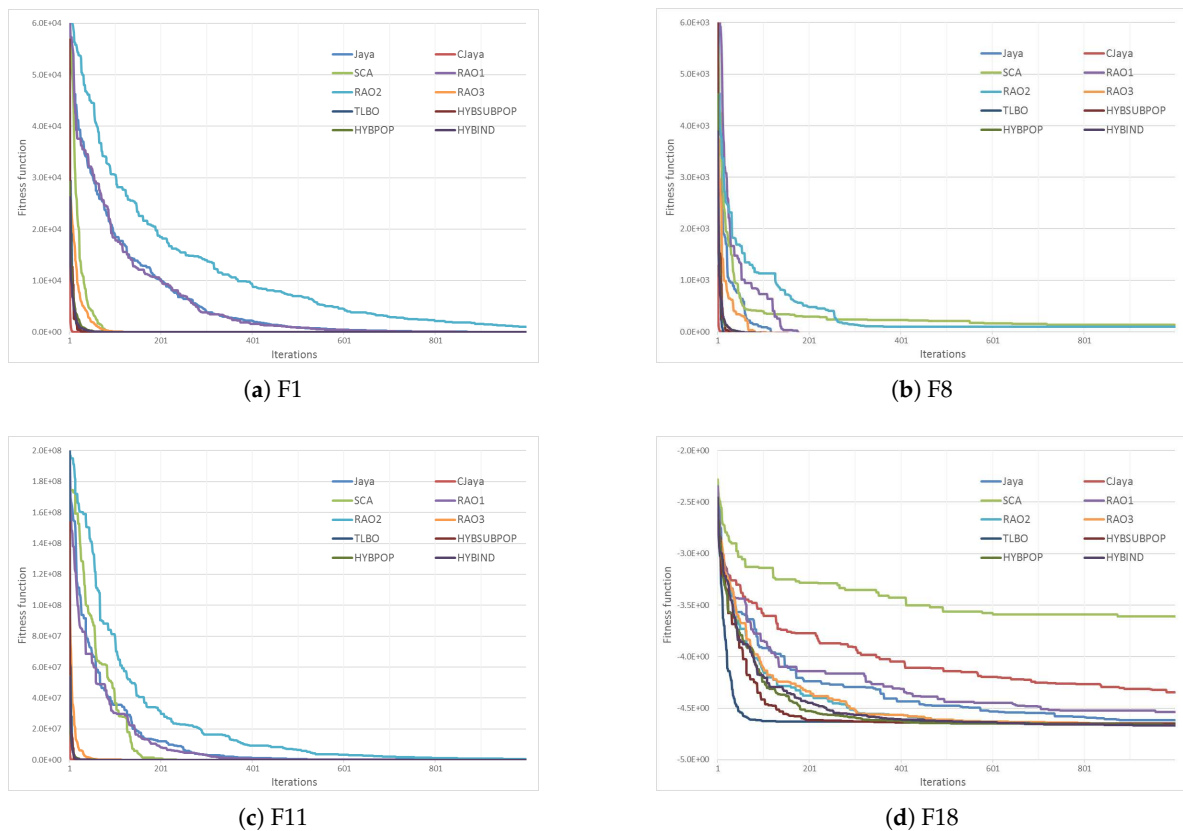
	SCA					RAO1				
	NoR -AI	NoR -BI	NoR -BwT	LtI -AI	LtI -BI	NoR -AI	NoR -BI	NoR -BwT	LtI -AI	LtI -BI
F1	209388 (323)	1397 (24)	1339 (23)	12387 (440)	4261 (181)	282204 (671)	4109 (35)	3847 (30)	49999 (0)	49985 (15)
F2	208824 (335)	1397 (16)	1344 (15)	12129 (232)	4222 (99)	282675 (1315)	4105 (47)	3867 (43)	49999 (0)	49990 (7)
F3	3263 (59)	24 (5)	11 (2)	49999 (0)	49908 (252)	10490 (104)	71 (5)	59 (5)	1037 (122)	937 (124)
F4	3522 (71)	30 (5)	10 (3)	49999 (0)	49983 (27)	5921 (61)	41 (5)	23 (3)	819 (431)	588 (79)
F5	81841 (285)	631 (25)	622 (24)	4113 (89)	2540 (177)	102605 (355)	739 (23)	728 (23)	6077 (21)	5996 (23)
F6	6613 (158)	42 (6)	0 (0)	49999 (0)	49997 (2)	20964 (211)	154 (8)	111 (6)	45874 (1909)	41787 (2043)
F7	7416 (89)	60 (7)	0 (0)	49999 (0)	49999 (0)	14501 (132)	114 (9)	57 (6)	48669 (1124)	19652 (15645)
F8	11411 (197)	90 (12)	0 (0)	49999 (0)	49998 (0)	21868 (505)	190 (14)	77 (4)	49782 (242)	23351 (13762)
F9	137414 (296)	1134 (26)	1098 (24)	13643 (141)	7062 (415)	394478 (748)	3436 (58)	3350 (58)	49992 (15)	49865 (102)
F10	131216 (572)	1416 (18)	1352 (18)	20314 (684)	13659 (992)	50688 (792)	573 (13)	299 (16)	49999 (0)	49912 (58)
F11	22671 (509)	173 (11)	0 (0)	49999 (0)	49998 (1)	49031 (5791)	730 (88)	116 (56)	49986 (15)	49824 (147)
F12	7116 (232)	59 (8)	8 (3)	49999 (0)	49996 (5)	18005 (6110)	144 (51)	97 (49)	15151 (12276)	9236 (6218)
F13	4083 (70)	37 (7)	0 (0)	49999 (0)	49996 (3)	1174 (2874)	9 (21)	0 (0)	9811 (24036)	8799 (21554)
F14	3161 (53)	22 (6)	0 (0)	49999 (0)	49985 (36)	2342 (185)	18 (3)	0 (0)	45878 (4245)	28668 (18831)
F15	5976 (33)	43 (6)	22 (5)	200 (19)	66 (6)	6468 (87)	46 (5)	24 (5)	263 (28)	217 (9)
F16	3314 (57)	27 (3)	10 (2)	49999 (0)	49998 (1)	10022 (135)	70 (11)	53 (9)	352 (9)	298 (4)
F17	3030 (30)	24 (2)	0 (0)	49999 (0)	49994 (9)	5717 (80)	39 (5)	1 (1)	4279 (749)	221 (12)
F18	5461 (100)	45 (4)	0 (0)	49999 (0)	49998 (2)	12198 (76)	100 (11)	0 (0)	47566 (3038)	7285 (1664)
F19	6201 (71)	41 (7)	22 (2)	259 (30)	68 (4)	6648 (89)	45 (5)	25 (3)	318 (12)	268 (7)
F20	5751 (66)	42 (3)	23 (4)	493 (46)	101 (10)	6677 (88)	42 (5)	25 (6)	10328 (19703)	389 (26)
F21	3330 (45)	26 (4)	12 (3)	49999 (0)	48236 (4088)	6390 (126)	48 (5)	31 (3)	41485 (6441)	22033 (13841)
F22	2986 (76)	25 (6)	0 (0)	49999 (0)	49975 (37)	1931 (162)	13 (2)	0 (0)	49470 (805)	24078 (13173)
F23	3699 (97)	29 (6)	0 (0)	49999 (0)	49951 (109)	6505 (2170)	53 (17)	0 (0)	670 (379)	215 (72)
F24	22109 (3516)	128 (7)	62 (4)	26470 (9520)	4247 (6275)	45468 (342)	441 (15)	217 (15)	8818 (7583)	4045 (500)
F25	23889 (117)	206 (12)	0 (0)	49999 (0)	49998 (1)	59024 (431)	843 (18)	618 (11)	10895 (61)	10755 (61)
F26	3205 (42)	21 (4)	0 (0)	49999 (0)	49958 (48)	5850 (130)	41 (6)	0 (0)	18757 (11293)	17964 (11291)
F27	5376 (621)	47 (8)	0 (1)	49999 (0)	49997 (2)	6280 (4993)	47 (36)	10 (23)	32659 (10816)	14141 (8996)
F28	5969 (241)	42 (3)	0 (0)	49999 (0)	49997 (2)	4077 (1375)	33 (14)	0 (0)	45602 (4353)	30802 (13495)

**Table 5.** Analysis of RAO2, RAO3, and teaching-learning-based optimization (TLBO) on function F1-F28 with a population size of 140.

	RAO2					RAO3					TLBO				
	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI
F1	158207 (321)	1833 (35)	1582 (25)	49999 (0)	49975 (20)	402040 (1764)	6191 (59)	5990 (52)	12845 (171)	12588 (152)	317548 (2268)	6636 (62)	6432 (62)	1816 (5)	1806 (4)
F2	158651 (522)	1838 (36)	1600 (30)	49999 (0)	49964 (30)	400507 (1638)	6179 (77)	5990 (71)	12748 (140)	12545 (122)	314216 (1213)	6582 (82)	6388 (79)	1802 (5)	1792 (4)
F3	9855 (93)	74 (11)	59 (9)	696 (40)	610 (45)	9933 (111)	74 (6)	60 (6)	665 (32)	583 (32)	10098 (209)	72 (7)	60 (7)	296 (81)	166 (8)
F4	5435 (79)	44 (9)	25 (6)	2038 (899)	2015 (897)	5509 (75)	44 (6)	25 (5)	1088 (614)	1063 (616)	5455 (124)	48 (5)	25 (4)	127 (19)	83 (4)
F5	107111 (218)	736 (10)	727 (10)	8885 (81)	8769 (77)	115700 (411)	756 (26)	744 (24)	12296 (180)	12161 (190)	151056 (1162)	764 (26)	751 (25)	1261 (5)	1229 (6)
F6	16646 (542)	129 (12)	93 (8)	49995 (3)	49669 (402)	16154 (615)	127 (11)	86 (10)	49984 (20)	49810 (159)	46797 (3371)	185 (17)	136 (7)	5463 (677)	3573 (250)
F7	11737 (91)	108 (5)	56 (5)	48902 (865)	7854 (5216)	11421 (172)	115 (8)	61 (3)	48508 (2285)	13021 (9489)	13459 (501)	139 (12)	70 (9)	49638 (279)	5639 (1539)
F8	17869 (1951)	218 (45)	63 (32)	49543 (413)	19788 (5571)	17255 (574)	188 (15)	76 (9)	48896 (1466)	25615 (9085)	202435 (46668)	3127 (779)	1418 (176)	49440 (444)	24482 (14497)
F9	158199 (509)	1411 (41)	1324 (38)	49999 (0)	49965 (34)	209334 (765)	2857 (50)	2785 (50)	17784 (301)	17582 (277)	277040 (1370)	3261 (53)	3180 (55)	2985 (11)	2958 (12)
F10	3889 (267)	31 (4)	0 (0)	49943 (52)	46966 (2679)	58363 (4038)	716 (60)	536 (62)	49998 (1)	49889 (103)	409579 (1597)	5276 (55)	5097 (60)	8337 (65)	8270 (71)
F11	34536 (5740)	524 (107)	8 (7)	49940 (79)	49106 (790)	47480 (2960)	905 (71)	7 (8)	49984 (17)	49761 (197)	4325394 (204520)	65328 (3160)	26532 (3689)	42857 (3655)	38436 (2036)
F12	19104 (215)	187 (13)	136 (12)	1472 (917)	1393 (85)	12595 (5136)	123 (54)	57 (66)	6960 (2906)	3570 (619)	25139 (3571)	234 (11)	172 (17)	393 (190)	299 (161)
F13	640 (44)	9 (2)	0 (0)	49680 (301)	21886 (9803)	1285 (794)	14 (5)	0 (0)	49649 (532)	30857 (13644)	5251 (368)	49 (10)	0 (0)	48857 (1119)	1302 (467)
F14	4472 (752)	30 (7)	0 (0)	4716 (684)	1414 (683)	4423 (767)	32 (7)	0 (0)	6115 (3259)	3465 (3260)	4862 (125)	33 (5)	0 (0)	2518 (225)	64 (11)
F15	6939 (126)	43 (5)	22 (3)	667 (166)	348 (31)	6474 (72)	43 (4)	23 (2)	319 (17)	275 (17)	8420 (320)	48 (5)	26 (3)	63 (1)	51 (1)
F16	9812 (85)	69 (8)	53 (6)	534 (29)	464 (20)	2904 (42)	22 (5)	7 (2)	49838 (177)	32614 (13514)	11723 (260)	69 (5)	54 (5)	103 (2)	86 (1)
F17	5089 (55)	40 (5)	1 (0)	2641 (726)	149 (7)	5103 (70)	40 (4)	1 (1)	2187 (425)	144 (9)	5888 (330)	35 (5)	1 (0)	2137 (713)	56 (7)
F18	9466 (106)	95 (8)	0 (0)	48411 (1305)	1938 (786)	9493 (136)	93 (6)	0 (0)	49151 (579)	2102 (570)	14358 (3227)	122 (23)	0 (0)	49208 (668)	1466 (866)

Table 5. Cont.

	RAO2					RAO3					TLBO				
	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI	NoR-AI	NoR-BI	NoR-BwT	LtI-AI	LtI-BI
F19	7104 (123)	49 (6)	27 (5)	833 (77)	547 (96)	6677 (99)	46 (4)	24 (4)	561 (225)	509 (220)	7812 (327)	43 (2)	21 (3)	73 (3)	58 (2)
F20	6847 (122)	42 (4)	23 (2)	3782 (894)	1677 (867)	5730 (77)	45 (7)	26 (5)	1115 (568)	1044 (569)	8677 (563)	47 (6)	25 (5)	103 (6)	75 (5)
F21	6889 (86)	45 (4)	28 (2)	49200 (1009)	10483 (4233)	2627 (82)	21 (4)	4 (1)	49803 (141)	33314 (10046)	6407 (348)	46 (8)	30 (6)	27592 (16851)	9567 (11094)
F22	2050 (134)	15 (3)	0 (0)	47791 (3154)	22625 (14711)	2381 (502)	23 (8)	0 (0)	49441 (729)	35495 (14282)	27875 (2436)	218 (147)	55 (1)	49997 (2)	47112 (3039)
F23	6113 (58)	49 (4)	0 (0)	241 (19)	151 (3)	6095 (67)	52 (5)	0 (0)	249 (26)	147 (4)	7185 (323)	57 (6)	0 (0)	149 (27)	57 (2)
F24	38389 (18515)	371 (12)	182 (91)	32501 (12156)	15916 (12065)	27517 (262)	355 (19)	177 (10)	1532 (987)	841 (36)	25248 (557)	402 (16)	201 (15)	237 (17)	167 (8)
F25	52014 (331)	758 (34)	554 (24)	4811 (210)	4799 (209)	49682 (265)	741 (33)	545 (25)	1584 (37)	1570 (35)	140740 (43787)	2360 (744)	837 (1381)	21162 (22046)	7098 (3282)
F26	5153 (121)	39 (5)	0 (0)	1061 (882)	883 (870)	5263 (242)	39 (5)	0 (0)	6302 (1120)	1436 (970)	6689 (956)	38 (7)	0 (0)	15868 (4085)	758 (373)
F27	9039 (372)	93 (5)	16 (27)	14656 (543)	4382 (5900)	9148 (289)	92 (5)	23 (28)	19748 (23981)	2674 (2360)	12647 (1424)	222 (63)	124 (42)	21151 (11703)	5807 (3567)
F28	3993 (525)	33 (8)	0 (0)	43661 (10700)	35225 (13441)	2318 (291)	18 (5)	0 (0)	49411 (561)	26332 (13590)	38285 (4443)	238 (13)	162 (19)	19500 (19909)	897 (345)

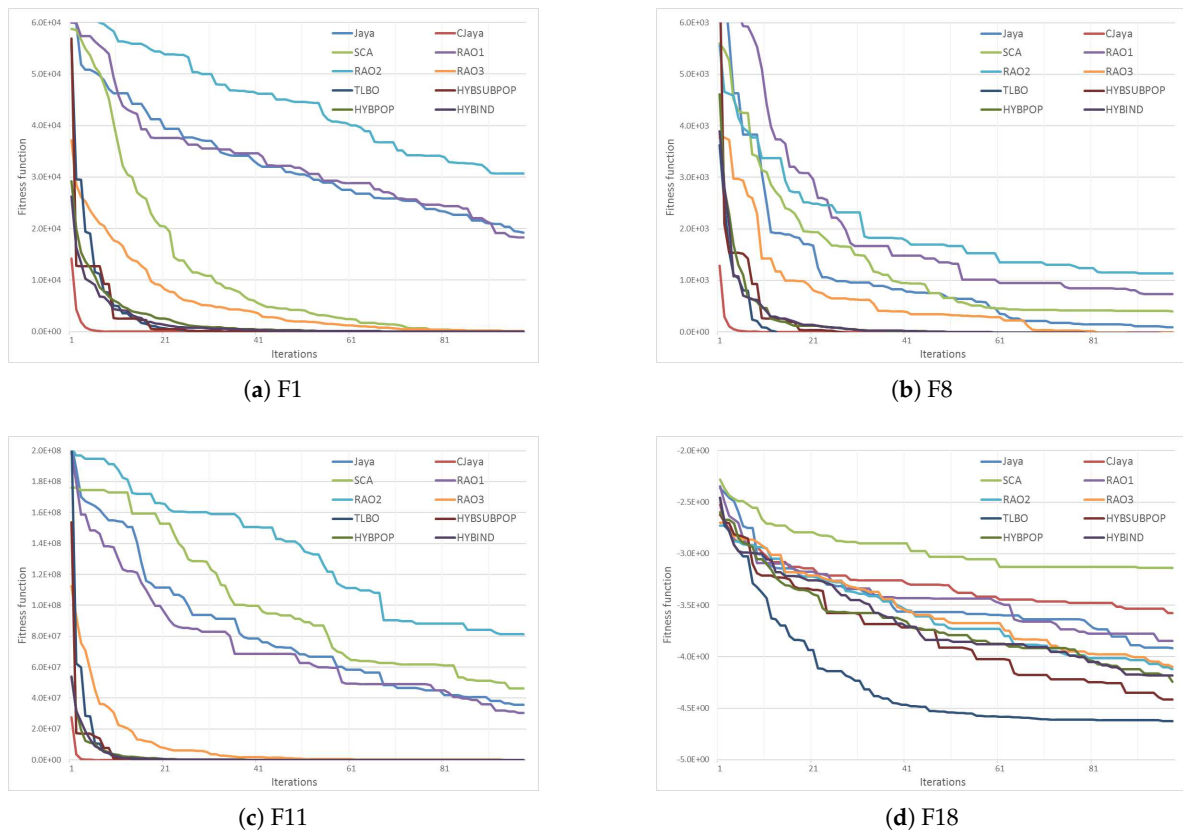


**Figure 1.** Convergence curves. The population size is set as 140 and the number of iterations to 1000.

**Table 6.** Ranking of the algorithms according to the number of iterations required to achieve an error of less than 0.001. The population size is set as 140.

	1	2	3	4	5	6	7	8	9	10
F1	CJaya	TLBO	HYBPOP	SCA	HYBIND	RAO3	HYBSUBPOP	RAO1	Jaya	RAO2
F2	CJaya	TLBO	HYBPOP	SCA	HYBIND	HYBSUBPOP	RAO3	RAO1	Jaya	RAO2
F3	TLBO	HYBPOP	CJaya	HYBSUBPOP	HYBIND	RAO3	SCA	RAO2	Jaya	RAO1
F4	CJaya	HYBPOP	TLBO	HYBSUBPOP	HYBIND	SCA	Jaya	RAO1	RAO2	RAO3
F5	HYBPOP	SCA	HYBIND	HYBSUBPOP	TLBO	RAO3	Jaya	RAO1	RAO2	CJaya
F6	TLBO	HYBPOP	Jaya	HYBIND	HYBSUBPOP	RAO3	RAO1	RAO2	SCA	CJaya
F7	TLBO	HYBPOP	RAO3	Jaya	RAO2	RAO1	HYBIND	HYBSUBPOP	SCA	CJaya
F8	HYBPOP	Jaya	RAO3	TLBO	RAO1	HYBIND	HYBSUBPOP	SCA		
F9	CJaya	HYBPOP	TLBO	HYBIND	HYBSUBPOP	SCA	RAO3	RAO1	Jaya	RAO2
F10	CJaya	HYBPOP	TLBO	HYBIND	HYBSUBPOP	SCA	RAO3	RAO1		
F11	HYBPOP	Jaya	RAO1	TLBO	RAO3	RAO2				
F12	TLBO	RAO2	RAO1	HYBIND	Jaya	HYBSUBPOP	HYBPOP	SCA		
F13	HYBPOP	SCA	HYBIND	HYBSUBPOP	TLBO	Jaya				
F14	TLBO	CJaya	HYBSUBPOP	RAO2	HYBIND	HYBPOP	RAO3	Jaya	SCA	RAO1
F15	CJaya	HYBPOP	SCA	HYBSUBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2
F16	TLBO	HYBPOP	RAO1	CJaya	HYBSUBPOP	HYBIND	RAO2	RAO3	Jaya	SCA
F17	TLBO	HYBPOP	HYBIND	Jaya	HYBSUBPOP	RAO2	RAO3	RAO1	CJaya	SCA
F18	HYBIND	HYBSUBPOP	Jaya	RAO1	SCA	HYBPOP				
F19	SCA	HYBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2	CJaya	HYBSUBPOP
F20	SCA	HYBPOP	HYBSUBPOP	HYBIND	TLBO	Jaya	RAO1	RAO3	RAO2	CJaya
F21	CJaya	TLBO	HYBPOP	HYBIND	HYBSUBPOP	SCA	RAO1	RAO3	RAO2	Jaya
F22	TLBO	HYBSUBPOP	HYBPOP	HYBIND	RAO3	RAO2	Jaya	SCA	CJaya	RAO1
F23	SCA	HYBIND	HYBPOP	RAO2	CJaya	Jaya	RAO3	TLBO		
F24	CJaya	HYBPOP	TLBO	HYBIND	RAO3	HYBSUBPOP	SCA	Jaya	RAO1	
F25	RAO1	Jaya	RAO2	HYBPOP	RAO3	HYBIND	HYBSUBPOP	SCA		
F26	HYBPOP	HYBIND	HYBSUBPOP	TLBO	Jaya	SCA	RAO2	CJaya	RAO1	RAO3
F27	TLBO	HYBIND								
F28	TLBO	HYBPOP	HYBIND	HYBSUBPOP	SCA					





**Figure 2.** Convergence curves. The population size is set as 140 and the number of iterations to 100.

**Table 7.** Computational times (s.) for 50 runs. The population size is set as 140 and  $max\_ITs$  is set to 50,000.

	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO	HYBSUBPOP	HYBPOP	HYBIND
F1	275.0	3263.8	1355.0	178.0	306.5	785.6	1515.0	743.2	1346.7	856.2
F2	300.5	3050.8	1406.5	193.3	324.2	811.8	764.9	777.9	1394.7	861.1
F3	50.5	231.5	105.0	43.5	55.6	55.7	44.0	92.3	89.7	85.1
F4	48.5	233.4	120.9	41.8	53.1	60.6	45.9	104.1	91.2	94.9
F5	95.3	184.8	157.2	90.4	99.7	93.8	97.9	69.5	53.8	121.5
F6	39.5	381.4	137.3	25.0	49.6	49.6	25.7	97.7	107.7	97.2
F7	58.8	544.8	221.1	33.8	68.8	68.0	33.1	144.2	157.9	138.7
F8	94.6	898.2	370.2	58.5	108.8	108.7	52.6	238.5	258.1	228.1
F9	140.4	932.0	583.1	119.3	151.5	311.0	311.0	312.9	328.0	313.7
F10	411.4	3435.2	1732.1	291.7	444.8	425.5	748.5	889.7	1446.7	803.1
F11	291.9	2652.7	1023.1	182.3	315.4	313.5	153.9	688.9	752.0	661.2
F12	48.8	457.8	186.0	30.8	61.0	59.5	28.7	122.8	130.2	118.4
F13	1760.6	1933.4	1850.7	1638.7	1619.4	1766.0	1738.0	1745.4	2057.6	1801.4
F14	33.6	210.9	94.2	26.7	37.4	36.9	28.9	71.6	72.9	69.9
F15	34.9	208.1	91.9	26.6	39.4	38.5	28.6	99.8	71.0	68.0
F16	20.2	187.3	75.8	12.7	25.6	28.3	15.1	56.2	54.9	52.4
F17	93.3	307.2	247.7	87.4	101.9	98.8	97.2	212.3	165.1	159.9
F18	295.7	763.3	615.6	281.3	296.2	296.0	263.3	683.1	442.5	472.1
F19	31.5	200.4	89.5	23.9	36.9	32.5	26.2	89.6	64.6	62.4
F20	31.8	198.4	79.0	22.1	37.7	36.4	23.4	75.2	63.8	60.2
F21	23.2	196.0	79.1	15.1	31.6	31.0	17.4	60.9	58.1	57.9
F22	305.9	650.5	443.7	298.8	319.5	312.6	295.2	388.9	426.4	382.9
F23	79.7	342.2	166.2	67.6	85.1	89.3	66.6	126.4	138.3	128.4
F24	197.3	1381.3	620.6	143.5	351.1	205.8	124.8	618.4	427.2	433.5
F25	4336.8	3192.1	1694.5	4063.5	4614.0	4018.7	1470.0	2253.0	4266.2	1493.1
F26	161.7	360.4	244.9	153.7	166.4	167.2	159.9	209.7	226.8	206.5
F27	191.5	651.4	361.7	198.1	194.4	197.9	174.5	306.0	316.4	298.1
F28	561.6	969.3	697.7	533.4	553.5	559.0	514.1	638.8	698.8	617.7

**Table 8.** Contribution of each algorithm to the replacements of the individuals (NoR-AI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	25125 (224)	14555 (80)	96302 (832)	7 (3)	17725 (165)	332520 (3125)	111730 (566)	51946 (201)	9347 (683)	43566 (1647)	5 (2)	25863 (189)	4901 (709)	58819 (2781)
F2	25246 (153)	12585 (3435)	96155 (691)	9 (3)	17679 (187)	333194 (2382)	98347 (24920)	51874 (204)	9051 (2019)	43800 (1249)	6 (1)	25999 (196)	5621 (916)	57287 (8893)
F3	2873 (51)	156 (9)	182 (24)	323 (53)	2594 (53)	6666 (208)	6861 (120)	613 (46)	288 (25)	185 (14)	49 (10)	625 (50)	17344 (8810)	2543 (65)
F4	1368 (31)	232 (12)	511 (29)	21 (4)	1374 (40)	2348 (239)	3590 (87)	633 (43)	334 (20)	275 (18)	15 (4)	602 (41)	55789 (19009)	1321 (40)
F5	4483 (164)	1028 (134)	39631 (434)	33 (4)	4705 (189)	52409 (1294)	1444 (200)	11980 (71)	543 (219)	19085 (141)	29 (5)	12800 (54)	1346 (247)	15300 (170)
F6	1137 (58)	285 (14)	259 (34)	89 (14)	780 (43)	146009 (7004)	31692 (1833)	385 (73)	366 (40)	785 (256)	43 (8)	447 (62)	4170 (1343)	14094 (873)
F7	2909 (60)	253 (14)	135 (18)	873 (95)	3851 (56)	5720 (183)	5898 (117)	567 (44)	419 (29)	737 (116)	32 (9)	1195 (53)	60536 (14046)	8801 (1958)
F8	3570 (93)	369 (25)	180 (23)	840 (111)	4777 (125)	13342 (565)	15701 (760)	672 (50)	410 (39)	1112 (164)	22 (5)	1162 (89)	20321 (13307)	69118 (2852)
F9	9658 (145)	7123 (4361)	64165 (451)	116 (11)	2912 (67)	216548 (3530)	23519 (14231)	43762 (570)	4450 (2101)	28801 (120)	76 (11)	11319 (91)	4487 (871)	25089 (3527)
F10	1055 (73)	13634 (3278)	27550 (741)	51 (8)	198 (13)	535232 (7868)	109601 (25934)	2284 (233)	12827 (2881)	23845 (1573)	37 (7)	267 (35)	27714 (10418)	55571 (8180)
F11	4617 (1330)	536 (22)	834 (360)	818 (442)	3734 (1758)	124739 (38358)	301758 (42543)	1165 (52)	494 (13)	4006 (80)	3 (1)	530 (27)	5319 (1354)	123365 (553)
F12	3825 (1912)	393 (23)	421 (69)	488 (438)	5401 (1662)	6225 (3380)	19531 (4751)	611 (164)	482 (32)	1090 (32)	30 (7)	863 (176)	3911 (1281)	9347 (1420)
F13	43 (5)	409 (29)	1617 (71)	2 (1)	1530 (83)	8015 (1388)	4041 (512)	256 (16)	627 (39)	663 (25)	3 (1)	973 (35)	32210 (18471)	1423 (95)
F14	1100 (456)	91 (8)	239 (102)	114 (20)	985 (387)	542 (106)	3217 (75)	468 (101)	243 (37)	481 (39)	55 (10)	413 (73)	779 (107)	1319 (219)
F15	670 (35)	926 (100)	2581 (55)	38 (5)	651 (33)	3111 (109)	1274 (141)	784 (31)	498 (54)	1340 (29)	29 (4)	785 (29)	510 (68)	1209 (28)
F16	272 (15)	142 (9)	155 (14)	120 (18)	4817 (105)	10285 (384)	6404 (93)	201 (18)	400 (24)	246 (17)	45 (7)	920 (66)	45629 (10723)	2258 (46)
F17	1614 (53)	80 (4)	62 (9)	504 (28)	1609 (36)	2146 (162)	3008 (40)	559 (26)	195 (11)	168 (22)	51 (6)	511 (32)	69441 (12934)	1290 (42)
F18	2931 (747)	85 (7)	64 (9)	222 (67)	4076 (996)	4234 (1275)	10162 (2047)	351 (78)	219 (23)	906 (24)	52 (5)	461 (81)	1122 (132)	4273 (966)
F19	512 (54)	975 (63)	3020 (130)	38 (6)	489 (66)	2550 (291)	1368 (93)	818 (22)	508 (68)	1398 (22)	31 (7)	818 (22)	543 (28)	1252 (33)
F20	594 (32)	985 (66)	2289 (71)	38 (7)	531 (21)	4788 (171)	1270 (89)	857 (29)	500 (76)	1199 (41)	29 (7)	769 (30)	816 (103)	1199 (33)
F21	164 (10)	189 (14)	302 (27)	209 (18)	2708 (99)	8114 (480)	3782 (97)	143 (18)	531 (16)	281 (18)	37 (9)	1078 (139)	10618 (6726)	1440 (50)
F22	380 (23)	66 (5)	558 (40)	240 (25)	378 (41)	866 (85)	5278 (1041)	247 (27)	89 (14)	415 (18)	114 (17)	242 (25)	740 (47)	4710 (909)
F23	1885 (43)	62 (6)	45 (5)	689 (49)	2101 (42)	2698 (166)	3214 (55)	664 (54)	214 (24)	214 (17)	48 (9)	571 (36)	97302 (14611)	1593 (93)
F24	6959 (200)	1882 (24)	7941 (216)	27 (3)	5402 (151)	50267 (786)	7073 (106)	4512 (105)	1222 (18)	5623 (65)	24 (5)	2781 (63)	532 (116)	4525 (91)
F25	6308 (1141)	368 (23)	164 (19)	2774 (647)	21124 (4220)	28995 (2630)	29346 (4062)	774 (33)	737 (37)	4481 (74)	4 (3)	1835 (102)	7938 (1834)	82428 (9703)
F26	1524 (28)	122 (10)	306 (44)	146 (15)	1496 (35)	1971 (155)	3772 (322)	364 (181)	333 (38)	480 (132)	77 (17)	342 (162)	17529 (24235)	1563 (199)
F27	1488 (1027)	77 (6)	869 (802)	311 (138)	1909 (1379)	3090 (2155)	7708 (1192)	290 (46)	193 (47)	884 (51)	103 (13)	378 (95)	1158 (403)	3392 (613)
F28	184 (18)	100 (9)	1674 (225)	210 (37)	875 (130)	1065 (192)	27356 (4070)	165 (22)	316 (54)	1003 (37)	104 (10)	471 (71)	626 (62)	16520 (6926)

**Table 9.** Last iteration in which a replacement of any individual occurs (Ltl-AI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	27706 (133)	15952 (78)	27672 (116)	8 (4)	27687 (139)	27638 (111)	15974 (86)	41978 (123)	16719 (825)	17985 (439)	6 (3)	41958 (122)	12560 (642)	16529 (841)
F2	27598 (133)	14205 (3194)	27566 (131)	7 (4)	27585 (133)	27535 (144)	14220 (3210)	41855 (401)	16147 (3033)	18153 (471)	6 (3)	41851 (401)	12747 (940)	16037 (2917)
F3	1115 (53)	158 (43)	192 (60)	1077 (52)	1115 (50)	1091 (56)	1092 (54)	49975 (36)	36436 (7356)	48752 (3444)	171 (103)	49885 (142)	49959 (63)	1642 (67)
F4	668 (92)	138 (17)	485 (93)	138 (137)	671 (93)	660 (93)	533 (16)	49892 (207)	27089 (11247)	49998 (1)	53 (18)	49917 (191)	49998 (2)	570 (26)
F5	8136 (83)	301 (42)	8124 (76)	24 (13)	8146 (75)	8100 (70)	297 (41)	7280 (154)	267 (97)	6607 (109)	25 (10)	7281 (155)	4726 (627)	2744 (52)
F6	49889 (109)	1032 (448)	853 (186)	4730 (3277)	49915 (101)	49740 (493)	46411 (1886)	49059 (1626)	48335 (1169)	49999 (0)	1034 (907)	49763 (415)	49816 (230)	47269 (2201)
F7	42814 (5594)	120 (18)	80 (9)	41573 (9922)	45096 (3793)	45055 (4291)	40202 (4886)	49995 (4)	43638 (4013)	49999 (0)	59 (23)	49996 (2)	49998 (1)	46348 (3456)
F8	48715 (1000)	320 (47)	256 (52)	46062 (3367)	47122 (2979)	48564 (1758)	49177 (742)	49990 (9)	45549 (3461)	49999 (0)	16 (4)	49993 (4)	49997 (2)	49803 (290)
F9	30943 (525)	4732 (2638)	30905 (536)	67 (17)	30884 (536)	30842 (530)	4733 (2642)	49998 (1)	4623 (2084)	18878 (479)	65 (16)	49998 (1)	11271 (963)	6159 (1442)
F10	49434 (873)	27375 (6641)	49999 (0)	44 (14)	46704 (2947)	49999 (0)	27435 (6661)	49987 (26)	29386 (6347)	49999 (0)	45 (10)	49946 (62)	49780 (206)	29077 (5971)
F11	14618 (1006)	2561 (745)	40446 (19107)	13232 (8804)	14525 (13913)	49776 (563)	49999 (0)	49683 (135)	24945 (10461)	49999 (0)	7 (5)	49608 (197)	46959 (426)	49999 (0)
F12	9357 (7181)	302 (45)	5310 (81)	5055 (4578)	10953 (6044)	14699 (14643)	9658 (5204)	49072 (1526)	43353 (6153)	49999 (0)	443 (178)	49822 (375)	49922 (190)	8679 (13743)
F13	14277 (2793)	1063 (102)	14850 (2653)	1830 (5416)	46409 (3263)	47589 (3459)	46804 (4071)	16920 (2372)	49248 (721)	49999 (0)	703 (690)	49922 (70)	49998 (1)	39259 (8692)
F14	30447 (20306)	174 (88)	30620 (22023)	627 (873)	30428 (20332)	27404 (19241)	2988 (1819)	49507 (780)	43442 (5080)	49999 (0)	359 (768)	49513 (782)	49870 (240)	3924 (6743)
F15	303 (6)	181 (6)	302 (6)	37 (17)	303 (7)	297 (7)	183 (10)	308 (20)	167 (9)	295 (25)	28 (21)	307 (19)	285 (22)	184 (7)
F16	267 (60)	82 (15)	110 (21)	157 (43)	1631 (72)	1527 (71)	863 (20)	25143 (13492)	48481 (1381)	49996 (6)	64 (17)	49897 (145)	49970 (81)	888 (34)
F17	4402 (1252)	72 (18)	64 (15)	2665 (595)	4278 (1104)	3223 (729)	3444 (707)	49989 (13)	30269 (7299)	49998 (0)	75 (24)	49987 (17)	49998 (1)	16011 (5032)
F18	44394 (5789)	6650 (12519)	288 (205)	43185 (5242)	48625 (986)	46910 (1507)	46650 (3721)	49367 (677)	45600 (4301)	49999 (0)	736 (1541)	49354 (658)	49679 (319)	26117 (11794)
F19	385 (27)	195 (13)	375 (31)	30 (6)	380 (26)	359 (25)	194 (5)	479 (101)	174 (16)	447 (97)	37 (26)	474 (99)	435 (94)	192 (6)
F20	663 (34)	211 (21)	640 (30)	39 (18)	653 (16)	615 (21)	209 (14)	708 (62)	210 (25)	633 (70)	52 (32)	694 (66)	623 (70)	245 (17)
F21	425 (69)	158 (30)	529 (106)	41474 (8300)	48520 (1006)	47210 (1836)	45157 (4091)	8538 (12362)	49320 (529)	49767 (439)	51 (27)	49443 (677)	49940 (85)	47027 (2504)
F22	44939 (4800)	15050 (11628)	49999 (0)	25381 (15057)	46460 (5448)	44796 (5873)	49990 (8)	47536 (2663)	44069 (4699)	49999 (0)	18746 (13839)	46949 (3518)	47932 (2883)	49985 (15)
F23	443 (35)	59 (13)	34 (4)	355 (29)	472 (52)	421 (26)	435 (40)	49995 (3)	39615 (9172)	49999 (0)	58 (29)	49991 (8)	49998 (1)	1276 (274)
F24	22208 (14958)	1723 (151)	25856 (1126)	11 (4)	13680 (13093)	4089 (85)	1431 (170)	17557 (14169)	1549 (127)	24593 (1456)	3479 (10405)	12641 (3699)	9469 (3695)	43818 (7627)
F25	12972 (13910)	499 (468)	1683 (66)	8051 (6134)	13175 (14209)	11759 (11152)	12305 (13043)	49942 (56)	37071 (2584)	49999 (0)	10730 (16943)	49992 (5)	49990 (10)	43810 (8205)
F26	4182 (1571)	2569 (1527)	3561 (1527)	2341 (1990)	4211 (1552)	4082 (1569)	3674 (1588)	46010 (7701)	45348 (6555)	49999 (0)	1303 (2433)	47091 (3918)	49816 (277)	13990 (9456)
F27	26951 (17884)	344 (92)	25400 (18353)	9212 (3658)	23800 (16455)	30524 (17310)	31150 (16665)	43480 (6458)	44977 (6360)	49999 (0)	9953 (14823)	47560 (2370)	46582 (4419)	27162 (21555)
F28	20387 (12935)	641 (559)	49999 (0)	7204 (3331)	45865 (4309)	47674 (1666)	42103 (14129)	23923 (9779)	48894 (900)	49999 (0)	2436 (4258)	43178 (5956)	45987 (2088)	45473 (13321)

Finally, Table 10 shows the last iteration in which each algorithm obtains a new optimum. A careful analysis of the results in Table 10 reveals that in the HYBPOP algorithm, the seven algorithms contribute similarly to reaching a better solution as new populations are produced. By contrast, when using the HYBIND algorithm, the powerful algorithms are CJaya and TLBO. It should be noted that the CJaya algorithm extracts random individuals from the population to generate new individuals. The TLBO algorithm collects all the individuals of the population to obtain new individuals. Therefore, these algorithms exploit the results obtained from the rest of the algorithms

to converge towards the optimum. This fact is due to the nature of these algorithms, where the best solution correctly guided the individuals. The mean value of the standard deviation is high because the LtI-BI is strongly affected by randomness behavior.

**Table 10.** Last iteration in which a replacement of the best individual occurs (LtI-BI). The population size is set as 140.

	HYBPOP							HYBIND						
	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO	Jaya	CJaya	SCA	RAO1	RAO2	RAO3	TLBO
F1	0 (0)	15863 (84)	0 (0)	0 (0)	0 (0)	0 (0)	15730 (116)	0 (0)	15946 (1848)	14473 (4866)	0 (0)	0 (0)	0 (0)	15509 (2075)
F2	0 (0)	12473 (5550)	0 (0)	0 (0)	0 (0)	0 (0)	12344 (5609)	0 (0)	15348 (4498)	12873 (6459)	0 (0)	0 (0)	0 (0)	15188 (4560)
F3	475 (382)	33 (31)	2 (2)	408 (367)	329 (404)	765 (275)	943 (55)	6 (15)	16 (13)	7 (7)	2 (6)	0 (0)	31 (26)	1288 (48)
F4	1 (2)	48 (27)	19 (18)	7 (11)	0 (0)	16 (19)	409 (16)	0 (0)	27 (15)	27 (18)	4 (10)	0 (0)	26 (26)	450 (20)
F5	0 (0)	38 (33)	10 (15)	0 (0)	0 (0)	0 (0)	17 (19)	0 (0)	68 (62)	21 (20)	0 (0)	0 (0)	4 (7)	8 (13)
F6	24 (41)	96 (143)	6 (19)	3 (5)	21 (42)	6850 (2653)	32134 (1169)	4 (9)	15 (14)	72 (93)	39 (89)	0 (0)	85 (112)	35901 (1651)
F7	1764 (1964)	23 (24)	11 (22)	1941 (2372)	1077 (1547)	5315 (7355)	5811 (8283)	0 (0)	21 (18)	2 (4)	10 (23)	2 (6)	46 (34)	24451 (11193)
F8	5681 (8188)	78 (33)	0 (0)	8600 (9945)	8735 (12438)	13766 (13950)	18316 (11867)	0 (0)	45 (16)	0 (0)	0 (0)	0 (0)	193 (329)	44652 (5431)
F9	0 (0)	3322 (2657)	0 (0)	0 (0)	0 (0)	1 (2)	3241 (2667)	0 (0)	3308 (1939)	0 (0)	3 (5)	0 (0)	1 (0)	3086 (1971)
F10	0 (0)	25452 (9341)	0 (0)	0 (0)	0 (0)	1 (2)	25355 (9451)	0 (0)	27404 (7900)	7926 (12114)	0 (0)	0 (0)	0 (0)	27330 (7991)
F11	10586 (11106)	1241 (809)	0 (0)	12384 (14485)	8518 (8532)	29734 (6041)	49977 (28)	0 (0)	357 (109)	0 (0)	0 (0)	0 (0)	0 (0)	49996 (4)
F12	750 (753)	84 (40)	22 (44)	867 (760)	741 (793)	758 (752)	2766 (812)	17 (49)	125 (96)	16 (32)	84 (128)	20 (60)	250 (101)	4035 (679)
F13	1 (1)	330 (158)	71 (86)	1 (1)	1 (3)	3 (4)	8260 (5143)	1 (1)	476 (228)	374 (162)	2 (5)	1 (1)	2 (3)	14327 (8720)
F14	11 (22)	5 (7)	0 (0)	1 (2)	1 (2)	5 (9)	414 (45)	0 (0)	5 (11)	2 (3)	1 (1)	1 (1)	6 (7)	560 (126)
F15	0 (0)	62 (50)	10 (17)	0 (0)	0 (0)	0 (0)	25 (31)	0 (0)	41 (28)	22 (35)	0 (0)	0 (0)	2 (3)	6 (8)
F16	1 (2)	29 (23)	4 (8)	23 (39)	9 (12)	19 (20)	705 (6)	1 (1)	14 (10)	2 (3)	8 (24)	1 (1)	23 (16)	753 (35)
F17	168 (96)	5 (13)	0 (0)	154 (70)	114 (97)	195 (57)	239 (15)	2 (4)	1 (1)	0 (0)	0 (0)	1 (1)	6 (8)	645 (188)
F18	6493 (12607)	6472 (12609)	5 (14)	6667 (12188)	8367 (11997)	7576 (12953)	8274 (12092)	3 (6)	2 (4)	0 (0)	534 (1594)	8 (18)	3 (4)	14666 (12688)
F19	1 (1)	59 (42)	8 (9)	0 (0)	0 (0)	2 (3)	11 (21)	0 (0)	66 (42)	51 (45)	0 (0)	0 (0)	6 (10)	12 (23)
F20	0 (0)	49 (43)	1 (1)	0 (0)	0 (0)	1 (3)	37 (41)	0 (0)	61 (47)	34 (22)	0 (0)	0 (0)	0 (0)	37 (58)
F21	1 (3)	47 (24)	8 (13)	8820 (5982)	6337 (15046)	9098 (16086)	5499 (12564)	1 (1)	32 (26)	10 (15)	2 (4)	0 (0)	4850 (14515)	16989 (17876)
F22	88 (164)	276 (827)	48 (104)	55 (84)	19 (42)	197 (166)	44183 (12373)	3 (8)	0 (0)	0 (0)	74 (183)	39 (81)	99 (139)	46093 (7310)
F23	211 (36)	5 (7)	0 (0)	177 (64)	151 (87)	213 (46)	247 (11)	1 (2)	8 (10)	0 (0)	6 (12)	1 (1)	3 (4)	564 (46)
F24	0 (0)	1034 (74)	0 (0)	0 (0)	0 (0)	0 (0)	786 (203)	0 (0)	1144 (85)	0 (0)	0 (0)	0 (0)	0 (0)	757 (177)
F25	3665 (679)	232 (134)	0 (0)	3948 (312)	3720 (479)	3687 (456)	3792 (485)	0 (0)	116 (45)	4662 (13984)	10704 (16898)	0 (0)	0 (0)	43752 (8269)
F26	26 (32)	59 (52)	2 (4)	6 (14)	9 (11)	5 (8)	632 (141)	11 (25)	59 (67)	9 (20)	11 (22)	10 (19)	20 (36)	900 (422)
F27	501 (919)	16 (32)	5004 (14995)	328 (483)	350 (477)	2802 (5567)	13485 (17667)	4058 (12128)	78 (87)	20001 (24493)	27 (71)	253 (752)	2746 (8058)	11151 (16657)
F28	2 (2)	36 (67)	2 (3)	18 (22)	4 (7)	9 (11)	11474 (8711)	0 (0)	13 (23)	3 (5)	30 (39)	11 (24)	16 (15)	11993 (7744)

It has been found that the HYBSUBPOP algorithm does not reach excellent optimization performance because of the lack of harmony between the original algorithms, so it has left without

further analysis. On the other hand, the exploitation phase of the HYBPOP and HYBIND algorithms are similar. In contrast, the HYBIND algorithm outperforms the HYBPOP one in terms of exploitation. The hybridization of the original algorithms is implemented at the individual level in the HYBIND algorithm, contrary to the HYBPOP algorithm, in which that hybridization is performed at the population level. Finally, the HYBPOP algorithm included algorithms that update the population without analyzing the fitness of the associated solutions, while this restriction is mandatory in the HYBIND algorithm.

## 5. Conclusions

This paper proposed a hybridization strategy of seven well-known algorithms. Three hybrid algorithms free of setting parameters dubbed the HYBSUBPOP, HYBPOP, and HYBIND algorithms are designed. These algorithms are derived from a dynamic skeleton allowing the inclusion of any metaheuristic optimization algorithm that exhibits further improvements. The only requirement in merging a new optimization algorithm into the proposed skeleton is to know if the replacement of an individual on that algorithm is based on the enhancement of the cost function or not. Moreover, both chaotic algorithms and multi-phase algorithms have been employed to design the proposed hybrid algorithms, which proves the versatility of the proposed hybridization skeleton. The experimental results show that the HYBPOP and HYBIND algorithms effectively exploit the capabilities of all the considered algorithms. They present an excellent ability to solve a large number of benchmark functions while improving the quality of the solutions obtained. Generally speaking, the hybridization at the individual level is better than that at the population level, which explains why the performance of the HYBSUBPOP algorithm is inferior to the other hybrid algorithms. As future lines of work, we intend to integrate more efficient algorithms into the proposed hybridization skeleton as well as to evaluate new versions of hybridization, and extend the performance analysis of the potential algorithms for solving more complex functions and real-world engineering problems.

**Author Contributions:** H.M., A.B., J.-L.S.-R., H.R., and A.J.-M. conceived the hybrid algorithms; A.B. conceived the Chaotic 2D Jaya algorithm; H.M. designed the hybrid algorithms; H.M. codified the hybrid algorithms; H.M., J.-L.S.-R., and H.R. performed numerical experiments; H.M., A.B., and A.J.-M. analyzed the data; H.M. wrote the original draft. A.B., J.-L.S.-R., H.R., and A.J.-M. reviewed and edited the manuscript. All the authors have read and agreed to the published version of the manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research and APC was funded by the Spanish Ministry of Science, Innovation and Universities and the Research State Agency under Grant RTI2018-098156-B-C54 co-financed by FEDER funds, and by the Spanish Ministry of Economy and Competitiveness under Grant TIN2017-89266-R, co-financed by FEDER funds.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine Blast Algorithm: A New Population Based Algorithm for Solving Constrained Engineering Optimization Problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612. doi:10.1016/j.asoc.2012.11.026.
2. Zhao, W.; Zhang, Z.; Wang, L. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Eng. Appl. Artif. Intell.* **2020**, *87*, 103300. doi:10.1016/j.engappai.2019.103300.
3. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. doi:10.1016/j.compstruc.2016.03.001.
4. Dorigo, M.; Di Caro, G. *New Ideas in Optimization*; Chapter The Ant Colony Optimization Meta-Heuristic; McGraw-Hill Ltd., UK: Maidenhead, UK, 1999; pp. 11–32.
5. Ma, H.; Simon, D.; Siarry, P.; Yang, Z.; Fei, M. Biogeography-Based Optimization: A 10-Year Review. *IEEE Trans. Emerg. Top. Comput. Intell.* **2017**, *1*, 391–407. doi:10.1109/TETCI.2017.2739124.
6. Ahrari, A.; Atai, A.A. Grenade Explosion Method—A novel tool for optimization of multimodal functions. *Appl. Soft Comput.* **2010**, *10*, 1132–1140. doi:10.1016/j.asoc.2009.11.032.

7. Poli, R.; Kennedy, J.; Blackwell, T. Particle swarm optimization. *Swarm Intell.* **2007**, *1*, 33–57. doi:10.1007/s11721-007-0002-0.
8. Xin-She, Y. Firefly Algorithm, Lévy Flights and Global Optimization. *Res. Dev. Intell. Syst.* **XXVI** **2009**, *4*, 209–218. doi:10.1007/978-1-84882-983-1\_15.
9. Karaboga, D.; Basturk, B. On the Performance of Artificial Bee Colony (ABC) Algorithm. *Appl. Soft Comput.* **2008**, *8*, 687–697. doi:10.1016/j.asoc.2007.05.007.
10. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. doi:10.1016/j.ins.2009.03.004.
11. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. doi:10.1080/03052150500384759.
12. Szénási, S.; Felde, I. Configuring genetic algorithm to solve the inverse heat conduction problem. *Acta Polytech. Hung.* **2017**, *14*, 133–152. doi:10.12700/APH.14.6.2017.6.8.
13. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. doi:10.1016/j.knosys.2015.12.022.
14. Rao, R.V.; Savsani, V.; Vakharia, D. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. doi:10.1016/j.cad.2010.12.015.
15. Zhao, W.; Wang, L.; Zhang, Z. Supply-Demand-Based Optimization: A Novel Economics-Inspired Algorithm for Global Optimization. *IEEE Access* **2019**, *7*, 73182–73206. doi:10.1109/ACCESS.2019.2918753.
16. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34. doi:10.5267/j.ijiec.2015.8.004.
17. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. doi:10.1016/j.future.2019.02.028.
18. Rao, R.V. Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *Int. J. Ind. Eng. Comput.* **2020**, *11*, 107–130. doi:10.5267/j.ijiec.2019.6.002.
19. Majumdar, M.; Mitra, T.; Nishimura, K. *Optim. Chaos*; Springer: New York, NY, USA, 2000.
20. Ott, E., Frontmatter. In *Chaos in Dynamical Systems*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2002; pp. i–iv.
21. Gandomi, A.; Yang, X.S.; Talatahari, S.; Alavi, A. Firefly algorithm with chaos. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 89–98. doi:10.1016/j.cnsns.2012.06.009.
22. Gokhale, S.; Kale, V. An application of a tent map initiated Chaotic Firefly algorithm for optimal overcurrent relay coordination. *Int. J. Electr. Power Energy Syst.* **2016**, *78*, 336–342. doi:10.1016/j.ijepes.2015.11.087.
23. Ma, Z.S. Chaotic populations in genetic algorithms. *Appl. Soft Comput.* **2012**, *12*, 2409–2424. doi:10.1016/j.asoc.2012.03.001.
24. Yan, X.F.; Chen, D.Z.; Hu, S.X. Chaos-genetic algorithms for optimizing the operating conditions based on RBF-PLS model. *Comput. Chem. Eng.* **2003**, *27*, 1393–1404. doi:10.1016/S0098-1354(03)00074-7.
25. Hong, W.C. Traffic flow forecasting by seasonal SVR with chaotic simulated annealing algorithm. *Neurocomputing* **2011**, *74*, 2096–2107. doi:10.1016/j.neucom.2010.12.032.
26. Mingjun, J.; Huanwen, T. Application of chaos in simulated annealing. *Chaos Solitons Fractals* **2004**, *21*, 933–941. doi:10.1016/j.chaos.2003.12.032.
27. Saremi, J.; Mirjalili, S.; Lewis, A. Biogeography-based optimisation with chaos. *Neural Comput. Appl.* **2014**, *25*, 1077–1097. doi:10.1007/s00521-014-1597-x.
28. Wang, X.; Duan, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Comput. Ind. Eng.* **2014**, *73*, 96–114. doi:10.1016/j.cie.2014.04.006.
29. Jia, D.; Zheng, G.; Khan, M.K. An effective memetic differential evolution algorithm based on chaotic local search. *Inf. Sci.* **2011**, *181*, 3175–3187. doi:10.1016/j.ins.2011.03.018.
30. Peng, C.; Sun, H.; Guo, J.; Liu, G. Dynamic economic dispatch for wind-thermal power system using a novel bi-population chaotic differential evolution algorithm. *Int. J. Electr. Power Energy Syst.* **2012**, *42*, 119–126. doi:10.1016/j.ijepes.2012.03.012.
31. Alatas, B. Chaotic bee colony algorithms for global numerical optimization. *Expert Syst. Appl.* **2010**, *37*, 5682–5687. doi:10.1016/j.eswa.2010.02.042.

32. Yu, J.; Kim, C.H.; Wadood, A.; Khurshid, T.; Rhee, S.B. A Novel Multi-Population Based Chaotic JAYA Algorithm with Application in Solving Economic Load Dispatch Problems. *Energies* **2018**, *11*. doi:10.3390/en11081946.
33. Farah, A.; Belazi, A. A novel chaotic Jaya algorithm for unconstrained numerical optimization. *Nonlinear Dyn.* **2018**, *93*, 1451–1480. doi:10.1007/s11071-018-4271-5.
34. Kumar, Y.; Singh, P.K. RA chaotic teaching learning based optimization algorithm for clustering problems. *Appl. Intell.* **2019**, *49*, 107–130. doi:10.1007/s10489-018-1301-4.
35. Moayedi, H.; Gör, M.; Khari, M.; Foong, L.K.; Bahiraei, M.; Bui, D.T. Hybridizing four wise neural-metaheuristic paradigms in predicting soil shear strength. *Measurement* **2020**, *156*, 107576. doi:10.1016/j.measurement.2020.107576.
36. Nguyen, B.M.; Tran, T.; Nguyen, T.; Nguyen, G. Hybridization of Galactic Swarm and Evolution Whale Optimization for Global Search Problem. *IEEE Access* **2020**, *8*, 74991–75010. doi:10.1109/ACCESS.2020.2988717.
37. Kayabekir, A.E.; Toklu, Y.C.; Bekdaş, G.; Nigdeli, S.M.; Yücel, M.; Geem, Z.W. A Novel Hybrid Harmony Search Approach for the Analysis of Plane Stress Systems via Total Potential Optimization. *Appl. Sci.* **2020**, *10*. doi:10.3390/app10072301.
38. Punurai, W.; Azad, M.S.; Pholdee, N.; Bureerat, S.; Sinsabvarodom, C. A novel hybridized metaheuristic technique in enhancing the diagnosis of cross-sectional dent damaged offshore platform members. *Comput. Intell.* **2020**, *36*, 132–150. doi:10.1111/coin.12247.
39. Pellegrini, R.; Serani, A.; Liuzzi, G.; Rinaldi, F.; Lucidi, S.; Diez, M. Hybridization of Multi-Objective Deterministic Particle Swarm with Derivative-Free Local Searches. *Mathematics* **2020**, *8*. doi:10.3390/math8040546.
40. Yue, Z.; Zhang, S.; Xiao, W. A Novel Hybrid Algorithm Based on Grey Wolf Optimizer and Fireworks Algorithm. *Sensors* **2020**, *20*. doi:10.3390/s20072147.
41. Seifi, A.; Ehteram, M.; Singh, V.P.; Mosavi, A. Modeling and Uncertainty Analysis of Groundwater Level Using Six Evolutionary Optimization Algorithms Hybridized with ANFIS, SVM, and ANN. *Sustainability* **2020**, *12*. doi:10.3390/su12104023.
42. Chen, X.; Yu, K. Hybridizing cuckoo search algorithm with biogeography-based optimization for estimating photovoltaic model parameters. *Sol. Energy* **2019**, *180*, 192–206. doi:10.1016/j.solener.2019.01.025.
43. Zhang, X.; Shen, X.; Yu, Z. A Novel Hybrid Ant Colony Optimization for a Multicast Routing Problem. *Algorithms* **2019**, *12*. doi:10.3390/a12010018.
44. Aljohani, T.M.; Ebrahim, A.F.; Mohammed, O. Single and Multiobjective Optimal Reactive Power Dispatch Based on Hybrid Artificial Physics–Particle Swarm Optimization. *Energies* **2019**, *12*. doi:10.3390/en12122333.
45. Ahmadian, A.; Elkamel, A.; Mazouz, A. An Improved Hybrid Particle Swarm Optimization and Tabu Search Algorithm for Expansion Planning of Large Dimension Electric Distribution Network. *Energies* **2019**, *12*. doi:10.3390/en12163052.
46. Li, G.; Liu, P.; Le, C.; Zhou, B. A Novel Hybrid Meta-Heuristic Algorithm Based on the Cross-Entropy Method and Firefly Algorithm for Global Optimization. *Entropy* **2019**, *21*. doi:10.3390/e21050494.
47. Cherki, I.; Chaker, A.; Djidar, Z.; Khalfallah, N.; Benzergua, F. A Sequential Hybridization of Genetic Algorithm and Particle Swarm Optimization for the Optimal Reactive Power Flow. *Sustainability* **2019**, *11*. doi:10.3390/su11143862.
48. hanem, W.A.H.M.; Jantan, A. Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. *Neural Comput. Appl.* **2018**, *30*, 163–181. doi:10.1007/s00521-016-2665-1.
49. Das, P.; Behera, H.; Panigrahi, B. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol. Comput.* **2016**, *28*, 14–28. doi:10.1016/j.swevo.2015.10.011.
50. Wang, G.G.; Gandomi, A.H.; Zhao, X.; Chu, H.C.E. Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. *Soft Comput.* **2016**, *20*, 273–285. doi:10.1007/s00500-014-1502-7.
51. Zhu, A.; Xu, C.; Li, Z.; Wu, J.; Liu, Z. Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. *J. Syst. Eng. Electron.* **2015**, *26*, 317–328. doi:10.1109/JSEE.2015.00037.
52. Javaid, N.; Ahmed, A.; Iqbal, S.; Ashraf, M. Day Ahead Real Time Pricing and Critical Peak Pricing Based Power Scheduling for Smart Homes with Different Duty Cycles. *Energies* **2018**, *11*. doi:10.3390/en11061464.

53. Mishra, S.; Ray, P.K. Power quality improvement using photovoltaic fed DSTATCOM based on JAYA optimization. *IEEE Trans. Sustain. Energy* **2016**, *7*, 1672–1680. doi:10.1109/TSTE.2016.2570256.
54. Huang, C.; Wang, L.; Yeung, R.S.; Zhang, Z.; Chung, H.S.; Bensoussan, A. A Prediction Model-Guided Jaya Algorithm for the PV System Maximum Power Point Tracking. *IEEE Trans. Sustain. Energy* **2018**, *9*, 45–55. doi:10.1109/TSTE.2017.2714705.
55. Abhishek, K.; Kumar, V.R.; Datta, S.; Mahapatra, S.S. Application of JAYA algorithm for the optimization of machining performance characteristics during the turning of CFRP (epoxy) composites: comparison with TLBO, GA, and ICA. *Eng. Comput.* **2016**, 1–19. doi:10.1007/s00366-016-0484-8.
56. Choudhary, A.; Kumar, M.; Unune, D.R. Investigating effects of resistance wire heating on AISI 1023 weldment characteristics during ASAW. *Mater. Manuf. Process.* **2018**, *33*, 759–769. doi:10.1080/10426914.2017.1415441.
57. Dinh-Cong, D.; Dang-Trung, H.; Nguyen-Thoi, T. An efficient approach for optimal sensor placement and damage identification in laminated composite structures. *Adv. Eng. Softw.* **2018**, *119*, 48–59. doi:10.1016/j.advengsoft.2018.02.005.
58. Singh, S.P.; Prakash, T.; Singh, V.; Babu, M.G. Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm. *Eng. Appl. Artif. Intell.* **2017**, *60*, 35–44. doi:10.1016/j.engappai.2017.01.008.
59. Cruz, N.C.; Redondo, J.L.; Álvarez, J.D.; Berenguel, M.; Ortigosa, P.M. A parallel Teaching–Learning–Based Optimization procedure for automatic heliostat aiming. *J. Supercomput.* **2017**, *73*, 591–606. doi:10.1007/s11227-016-1914-5.
60. Rao, R.V.; Pawar, R.B. Self-adaptive Multi-population Rao Algorithms for Engineering Design Optimization. *Appl. Artif. Intell.* **2020**, *34*, 187–250. doi:10.1080/08839514.2020.1712789.
61. Rao, R.; Pawar, R. Constrained design optimization of selected mechanical system components using Rao algorithms. *Appl. Soft Comput.* **2020**, *89*, 106141. doi:10.1016/j.asoc.2020.106141.
62. Rao, R.V.; Keesari, H.S. Rao algorithms for multi-objective optimization of selected thermodynamic cycles. *Eng. Comput.* **2020**. doi:10.1007/s00366-020-01008-9.
63. Kumar-Majhi, S. An Efficient Feed Forward Network Model with Sine Cosine Algorithm for Breast Cancer Classification. *Int. J. Syst. Dyn. Appl. (IJSDA)* **2018**, *7*, 1–14. doi:10.4018/IJSDA.2018040101.
64. Rajesh K.S., D.S. Load frequency control of autonomous power system using adaptive fuzzy based PID controller optimized on improved sine cosine algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 2361–2373. doi:10.1007/s12652-018-0834-z1.
65. Khezri, R.; Oshnoei, A.; Tarafdar Hagh, M.; Muyeen, S. Coordination of Heat Pumps, Electric Vehicles and AGC for Efficient LFC in a Smart Hybrid Power System via SCA-Based Optimized FOPID Controllers. *Energies* **2018**, *11*. doi:10.3390/en11020420.
66. Ramanaiah, M.L.; Reddy, M.D. Sine Cosine Algorithm for Loss Reduction in Distribution System with Unified Power Quality Conditioner. *i-Manag. J. Power Syst. Eng.* **2017**, *5*, 10–16. doi:10.26634/jps.5.3.13667.
67. Dhundhara, S.; Verma, Y.P. Capacitive energy storage with optimized controller for frequency regulation in realistic multisource deregulated power system. *Energy* **2018**, *147*, 1108–1128. doi:10.1016/j.energy.2018.01.076.
68. Singh, V.P. Sine cosine algorithm based reduction of higher order continuous systems. In Proceedings of the 2017 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 7–8 December 2017; pp. 649–653. doi:10.1109/ISS1.2017.8389252.
69. Das, S.; Bhattacharya, A.; Chakraborty, A.K. Solution of short-term hydrothermal scheduling using sine cosine algorithm. *Soft Comput.* **2018**, *22*, 6409–6427. doi:10.1007/s00500-017-2695-3.
70. Singh, M.; Panigrahi, B.; Abhyankar, A. Optimal coordination of directional over-current relays using Teaching Learning-Based Optimization (TLBO) algorithm. *Int. J. Electr. Power Energy Syst.* **2013**, *50*, 33–41. doi:10.1016/j.jepes.2013.02.011.
71. Niknam, T.; Azizipanah-Abarghooee, R.; Narimani, M.R. A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems. *Eng. Appl. Artif. Intell.* **2012**, *25*, 1577–1588. doi:10.1016/j.engappai.2012.07.004.
72. Li, D.; Zhang, C.; Shao, X.; Lin, W. A multi-objective TLBO algorithm for balancing two-sided assembly line with multiple constraints. *J. Intell. Manuf.* **2016**, *27*, 725–739. doi:10.1007/s10845-014-0919-2.



73. Arya, L.; Koshti, A. Anticipatory load shedding for line overload alleviation using Teaching learning based optimization (TLBO). *Int. J. Electr. Power Energy Syst.* **2014**, *63*, 862–877. doi:10.1016/j.ijepes.2014.06.066.
74. Mohanty, B. TLBO optimized sliding mode controller for multi-area multi-source nonlinear interconnected AGC system. *Int. J. Electr. Power Energy Syst.* **2015**, *73*, 872–881. doi:10.1016/j.ijepes.2015.06.013.
75. Yan, J.; Li, K.; Bai, E.; Yang, Z.; Foley, A. Time series wind power forecasting based on variant Gaussian Process and TLBO. *Neurocomputing* **2016**, *189*, 135–144. doi:10.1016/j.neucom.2015.12.081.
76. Baghban, A.; Kardani, M.N.; Mohammadi, A.H. Improved estimation of Cetane number of fatty acid methyl esters (FAMES) based biodiesels using TLBO-NN and PSO-NN models. *Fuel* **2018**, *232*, 620–631. doi:10.1016/j.fuel.2018.05.166.
77. Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.; Belazi, A. Efficient parallel and fast convergence chaotic Jaya algorithms. *Swarm Evol. Comput.* **2020**, 100698. doi:10.1016/j.swevo.2020.100698.
78. Ravipudi, J.L.; Neebha, M. Synthesis of linear antenna arrays using Jaya, self-adaptive Jaya and chaotic Jaya algorithms. *AEU-Int. J. Electron. Commun.* **2018**, *92*, 54–63. doi:10.1016/j.aeue.2018.05.022.
79. Free Software Foundation, Inc. GCC, the GNU Compiler Collection. Available online: <https://www.gnu.org/software/gcc/index.html> (accessed on 10th March 2017).
80. García-Monzó, A.; Migallón, H.; Jimeno-Morenilla, A.; Sánchez-Romero, J.L.; Rico, H.; Rao, R.V. Efficient Subpopulation Based Parallel TLBO Optimization Algorithms. *Electronics* **2018**, *8*. doi:10.3390/electronics8010019.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).